



CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Rashi Maharjan

London Met ID: 22067683

College ID: NP01CP4a220113

Group: L1C5

Assignment Due Date: Tuesday, May 9, 2023

Assignment Submission Date: Wednesday, May 10, 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. JAVA	1
1.2. BlueJ	1
2. Class Diagram and Inheritance Diagram	2
2.1 BankCard Class	2
2.2 DebitCard Class	3
2.3 CreditCard Class	4
2.4 BankGUI Class	5
3. Pseudo Code	7
3.1 BankGUI	7
4. Method descriptions	31
5. Inspection	33
<i>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.</i>	33
<i>Test 2: Evidences should be shown of: a. Add DebitCard b. Add CreditCard c. Withdraw amount from Debit card d. Set the credit limit e. Remove the credit card</i>	35
<i>Test 3: Test that appropriate dialog boxes appear when unsuitable values are entered for the Card ID, (include a screenshot of the dialog box, together with a corresponding screenshot of the GUI, showing the values that were entered).</i>	46
6. Error detection and Correction	48
Error 1: Syntax Error	48
Error 2: Semantic Error	48
Error 3: Logical Error	49
7. Conclusion	51
8. References	52
9. Appendix	53
i. Code of Bank Card	53
ii. Code of Debit Card	56
iii. Code of Credit Card	59
iv. Code of BankGUI	63

Table of Figures

FIGURE 1 CLASS DIAGRAM OF BANK CARD	2
FIGURE 2 CLASS DIAGRAM OF DEBIT CARD.....	3
FIGURE 3 CLASS DIAGRAM OF CREDIT CARD.....	4
FIGURE 4 CLASS DIAGRAM OF BANKGUI	5
FIGURE 5 INHERITANCE DIAGRAM BETWEEN BANK CARD, DEBIT CARD, CREDIT CARD AND BANKGUI	6
FIGURE 6 SCREENSHOT OF CLASS BEING COMPLIED	34
FIGURE 7 SCREENSHOT OF BANKGUI RUNNING.....	34
FIGURE 8 SCREENSHOT OF ADDING DEBIT CARD	35
FIGURE 9 SCREENSHOT OF CLICKING DISPLAY BUTTON WHEN ADDING DEBIT CARD	36
FIGURE 10 SCREENSHOT OF INFORMATION WHEN DEBIT CARD IS ADDED.	36
FIGURE 11 SCREENSHOT OF ADDING CREDIT CARD	37
FIGURE 12 SCREENSHOT OF CLICKING DISPLAY BUTTON WHEN ADDING CREDIT CARD.....	38
FIGURE 13 SCREENSHOT OF INFORMATION WHEN CREDIT CARD IS ADDED.	38
FIGURE 14 SCREENSHOT OF WITHDRAWING AMOUNT FROM DEBIT CARD	40
FIGURE 15 SCREENSHOT OF CLICKING DISPLAY BUTTON WHEN WITHDRAWING DEBIT CARD	40
FIGURE 16 SCREENSHOT OF INFORMATION WHEN DEBIT CARD IS WITHDRAWN	40
FIGURE 17 SCREENSHOT OF SETTING THE CREDIT LIMIT	42
FIGURE 18 SCREENSHOT OF CLICKING DISPLAY BUTTON WHEN SETTING THE CREDIT LIMIT	42
FIGURE 19 SCREENSHOT OF INFORMATION WHEN CREDIT LIMIT IS SET	43
FIGURE 20 SCREENSHOT OF REMOVING THE CREDIT CARD	45
FIGURE 21 SCREENSHOT OF THE DIALOG BOX WHICH APPEARED WHEN UNSUITABLE VALUES WERE ENTERED	47
FIGURE 22 SYNTAX ERROR.....	48
FIGURE 23 SYNTAX ERROR SOLUTION.....	48
FIGURE 24 SEMANTIC ERROR.....	48
FIGURE 25 SEMATIC ERROR SOLUTION	49
FIGURE 26 LOGICAL ERROR.....	49
FIGURE 27 LOGICAL ERROR SOLUTION	50

Table of Tables

TABLE 1 BANKGUI METHOD.....	32
TABLE 2 TEST THAT THE PROGRAM CAN BE COMPILED AND RUN USING THE COMMAND PROMPT.....	33
TABLE 3 ADDING DEBIT CARD	35
TABLE 4 ADDING OF CREDIT CARD	37
TABLE 5 WITHDRAW AMOUNT FROM CREDIT CARD.....	39
TABLE 6 SETTING THE CREDIT LIMIT	41
TABLE 7 REMOVE THE CREDIT CARD	44
TABLE 8 TESTING WHETHER THE DIALOG BOX APPEARS WHEN UNSUITABLE VALUES ARE ENTERED.....	46

1. Introduction

This project is the creation of bank card, debit cards, and credit cards using Java programming. The objectives of this project are designing and implementing classes and objects for bank cards, debit cards, and credit cards to make graphical user interface (GUI) for a system that stores details of Bank Card in an ArrayList. We will represent a number of card procedures, including cash withdrawal, account balance query, cancelation and granting of card as well as transaction history.

1.1. JAVA

Java is easy to use, object-oriented, distributed, interpretable, safe, neutral in terms of architecture, portable, high speed, multithreaded, and dynamic. The powerful, mission-critical apps can be created using Java, a full-featured, general-purpose programming language. The development of standalone apps for servers, desktop computers, and mobile devices is also done using it today, in addition to Web programming.

Today, Java is used frequently to create developing applications for Web servers. These programs handle data, carry out calculations, and produce dynamic Web pages. Java is a flexible programming language that may be used to create programs for desktop computers, servers, and compact mobile devices (Liang, 2012).

1.2. BlueJ

BlueJ is a language-based integrated development environment. Michael Kölling and John Rosenberg (Kölling & Rosenberg, 1996) created and implemented BlueJ to instruct object-oriented programming. In order to introduce Java to beginning programmers, BlueJ was originally adopted in 1999. The purpose of BlueJ is to make object-oriented idea learning easier. In this project, we have used BlueJ to carry out the JAVA programming (Hagan & Markham, 2000).

2. Class Diagram and Inheritance Diagram

A class diagram is a type of static structure diagram that depicts classes, constraints, and class properties. It provides a summary of a software system. An inheritance diagram depicts relationships between classes. It provides a high-level overview of the procedure for understanding user needs before designing, developing, and testing software that will meet those requirements (Walker & Matthew , 2022).

2.1 BankCard Class

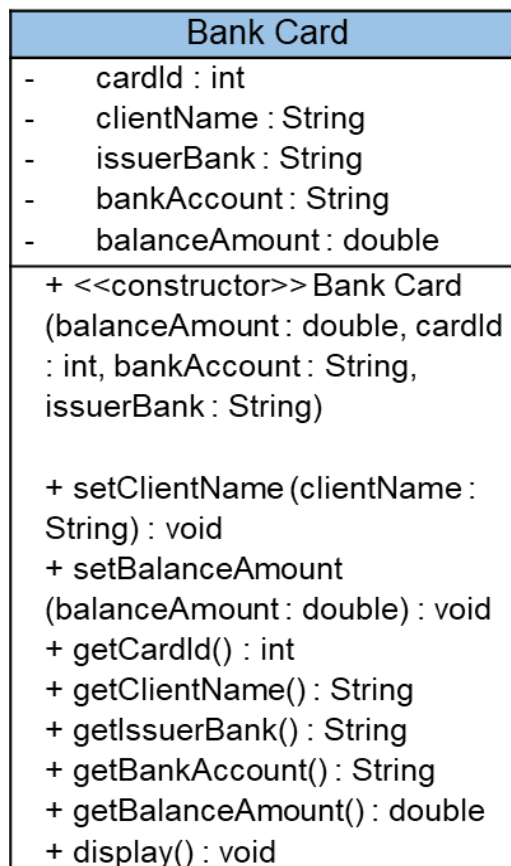


Figure 1 Class Diagram of Bank Card

2.2 DebitCard Class

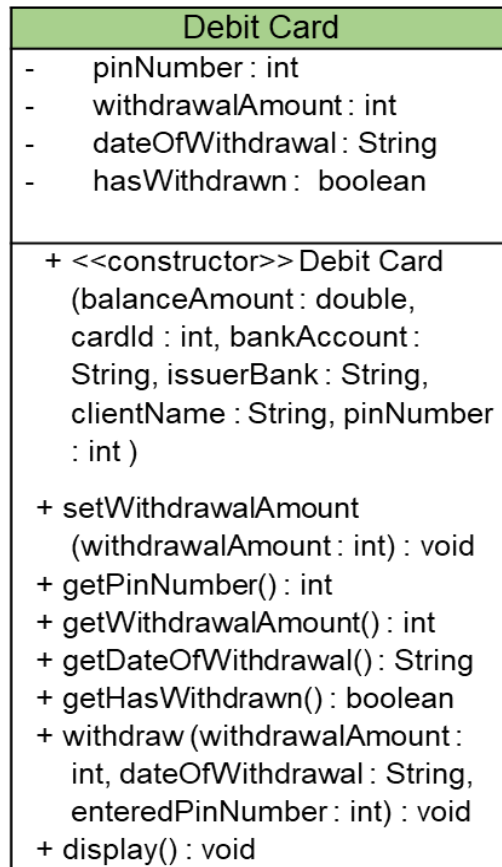


Figure 2 Class Diagram of Debit Card

2.3 CreditCard Class

Credit Card
<ul style="list-style-type: none"> - cvcNumber : int - creditLimit : double - interestRate : double - expirationDate : String - gracePeriod : int - isGranted : boolean
<p>+ <<constructor>> Credit Card (balanceAmount : double, cardId : int, bankAccount : String, issuerBank : String, clientName : String, cvcNumber : int, interestRate : double, expirationDate : String)</p> <p>+ setCreditLimit (creditLimit : double, gracePeriod : int) : void</p> <p>+ getCvcNumber() : int</p> <p>+ getInterestRate() : double</p> <p>+ getCreditLimit() : double</p> <p>+ getExpirationDate() : String</p> <p>+ getGracePeriod() : int</p> <p>+ getIsGranted() : boolean</p> <p>+ cancelCreditCard() : void</p> <p>+ display() : void</p>

Figure 3 Class Diagram of Credit Card

2.4 BankGUI Class

BankGUI
<ul style="list-style-type: none"> - frame: JFrame - TitleText, DebitText, CreditText – JLabel - D_CardIdText, D_ClientNameText, D_IssuerBankText, D_BankAccountText, D_BalanceAmountText, D_PinNumberText, D_CardIDText, D-WithdrawalAmountText, D_PinNumberrText – JTextField - D_CardIdLabel, D_ClientNameLabel, D_IssuerBankLabel, D_BankAccountLabel, D_BalanceAmountLabel, D_PinNumberLabel, D_CardIDLabel, D-WithdrawalAmountLabel, D-WithdrawalDateLabel, D_PinNumberrLabel – JLabel - D_DisplayButton, D_AddButton, D_DisplayyButton, D-WithdrawButton – JButton - day, month, year – JComboBox - C_CardIdText, C_ClientNameText, C_IssuerBankText, C_BankAccountText, C_BalanceAmountText, C_CVCNumberText, C_InterestRateText, C_CardIDText, C_CreditLimitText, C_GracePeriodText, C_CarDIDText – JTextField - C_CardIdLabel, C_ClientNameLabel, C_IssuerBankLabel, C_BankAccountLabel, C_ExpirationDateLabel, C_BalanceAmountLabel, C_CVCNumberLabel, C_InterestRateLabel, C_CardIDLabel, C_CreditLimitLabel, C_GracePeriodLabel, C_CarDIDLabel – JLabel - C_DisplayButton, C_AddButton, C_DisplayyButton, C_SetButton, C_DisplayyyButton, C_CancelButton – JButton - dayy, monthh, year – JComboBox - ClearButton- JButton + clientName, issuerBank, bankAccount, dateOfWithdrawal, expirationDate – String + cardId, balanceAmount, pinNumber, withdrawalAmount, cardId, pinNumbeR – int + cvcNumber, gracePeriod, carDID – int + creditLimit, interestRate- double + Existence_of_Card - boolean
<ul style="list-style-type: none"> + <<constructor>> BankGUI() + actionPerformed(ActionEvent e): void + main(String[] args): void

Figure 4 Class Diagram of BankGUI



Figure 5 Inheritance Diagram between Bank Card, Debit Card, Credit Card and BankGUI

3. Pseudo Code

Pseudo-code is an outline of a program that uses phrases that are typically used in spoken language and can easily be translated into actual programming statements. It is a method for describing computer programs that avoids utilizing the precise syntax and keywords of a programming language (Karatrantou & Panagiotakopoulos, 2008).

3.1 BankGUI

IMPORT javax.swing.*

IMPORT java.awt.event.*

IMPORT java.util.ArrayList

IMPORT java.awt.Color

CREATE a class BankGUI which implements ActionListener

DO

DECLARE instance variable frame as JFrame using private access modifier

DECLARE instance variable cards as ArrayList<BankCard>

DECLARE instance variables TitleText, DebitText, CreditText as JLabel using private access modifier

DECLARE instance variables D_CardIdText, D_ClientNameText, D_IssuerBankText, D_BankAccountText, D_BalanceAmountText as JTextField using private access modifier

DECLARE instance variables D_PinNumberText, D_CardIDText, D-WithdrawalAmountText, D_PinNumbererrText as JTextField using private access modifier

DECLARE instance variables D_CardIdLabel, D_ClientNameLabel, D_IssuerBankLabel, D_BankAccountLabel, D_BalanceAmountLabel as JLabel using private access modifier

DECLARE instance variables D_PinNumberLabel, D_CardIDLabel, D-WithdrawalAmountLabel, D-WithdrawalDateLabel, D_PinNumberrLabel as JLabel using private access modifier

DECLARE instance variables D_DisplayButton, D_AddButton, D_DisplayyButton, D-WithdrawButton as JButton using private access modifier

DECLARE instance variables day, month, year as JComboBox using private access modifier

DECLARE instance variables C_CardIdText, C_ClientNameText, C_IssuerBankText, C_BankAccountText, C_BalanceAmountText as JTextField using private access modifier

DECLARE instance variables C_CVCNumberText, C_InterestRateText, C_CardIDText, C_CreditLimitText, C_GracePeriodText, C_CarDIDText as JTextField using private access modifier

DECLARE instance variables C_CardIdLabel, C_ClientNameLabel, C_IssuerBankLabel, C_BankAccountLabel, C_ExpirationDateLabel, C_BalanceAmountLabel as JLabel using private access modifier

DECLARE instance variables C_CVCNumberLabel, C_InterestRateLabel, C_CardIDLabel, C_CreditLimitLabel, C_GracePeriodLabel, C_CarDIDLabel as JLabel using private access modifier

DECLARE instance variables C_DisplayButton, C_AddButton, C_DisplayyButton, C_SetButton, C_DisplayyyButton, C_CancelButton as JButton using private access modifier

DECLARE instance variables dayy, monthh, yearr as JComboBox using private access modifier

DECLARE instance variables ClearButton as JButton using private access modifier

CREATE a method BankGUI with return type void and no parameters

DO

INITIALIZE cards as ArrayList

INITIALIZE frame as JFrame passing "Nabil Bank"

SET the size of frame as (1280, 720)

SET the layout of frame as (null)

SET the Default Close Operation of frame as (JFrame.EXIT_ON_CLOSE)

SET the visibility of frame (true)

SET the resize of frame as (true)

DECLARE the required number of JLabel for Debit Card

DECLARE the required number of JTextField for Debit Card

DECLARE the required number of JButton for Debit Card

DECLARE the required number of JComboBox for Debit Card

DECLARE the required number of JLabel for Credit Card

DECLARE the required number of JTextField for Credit Card

DECLARE the required number of JButton for Credit Card

DECLARE the required number of JComboBox for Credit Card

SET the bounds for the required number of JLabel for Debit Card

SET the bounds for the required number of JTextField for Debit Card

SET the bounds for the required number of JButton for Debit Card

SET the bounds for the required number of JComboBox for Debit Card

SET the bounds for the required number of JLabel for Credit Card

SET the bounds for the required number of JTextField for Credit Card

SET the bounds for the required number of JButton for Credit Card

SET the bounds for the required number of JComboBox for Credit Card

ADD the required number of JLabel for Debit Card

ADD the required number of JTextField for Debit Card

ADD the required number of JButton for Debit Card

ADD the required number of JComboBox for Debit Card

ADD the required number of JLabel for Credit Card

ADD the required number of JTextField for Credit Card

ADD the required number of JButton for Credit Card

ADD the required number of JComboBox for Credit Card

END DO

CREATE a method actionPerformed with ActionEvent e as parameter and return type as void

DO

IF e.getSource() is D_AddButton

DO

IF D_CardIdText.getText() is empty or D_ClientNameText.getText() is empty or D_IssuerBankText.getText() is empty or D_BankAccountText.getText() is empty or D_BalanceAmountText.getText() is empty or D_PinNumberText.getText() is empty

DO

DISPLAY "The fields are empty!!!!" from a MessageDialog

END DO

ELSE

DO

TRY

DO

DECLARE clientName as string with value of D_ClientNameText.getText()

DECLARE issuerBank as string with value of D_IssuerBankText.getText()

DECLARE bankAccount as string with value of D_BankAccountText.getText()

DECLARE cardId as int with value of
D_CardIdText.getText()

DECLARE balanceAmount as int with value of
D_BalanceAmountText.getText()

DECLARE pinNumber as int with value of
D_PinNumberText.getText()

DECLARE Existence_of_Card as boolean with the value of
false

FOR BankCard DebitCard from cards

DO

IF DebitCard is an instance of DebitCard

DO

DOWNCAST DebitCard to DebitCard and
store it to DebitCard_Object of DebitCard

IF DebitCard_Object.getCardId() is equal to
cardId

DO

DECLARE Existence_of_Card as true

END DO

END DO

END DO

IF Existence_of_Card is equal to false

DO

CREATE a new DebitCard object with parameters
(balanceAmount, cardId, bankAccount, issuerBank,
clientName, pinNumber)

ADD DebitCard_Object to the end of the cards list

DISPLAY "Your Debit Card has been added
successfully!!!" from a MessageDialog

```

        ELSE
        DO
            DISPLAY "Debit Card already exists!!!" from a
            MessageBox
        END DO
    END DO
    CATCH NumberFormatException nfe
    DO
        DISPLAY "Please provide required information only!!!" from
        a MessageBox
    END DO
END DO

IF e.getSource() is D_DisplayButton
DO
    IF D_CardIdText.getText() is empty or D_ClientNameText.getText() is
    empty or D_IssuerBankText.getText() is empty or
    D_BankAccountText.getText() is empty or
    D_BalanceAmountText.getText() is empty or D_PinNumberText.getText()
    is empty
    DO
        DISPLAY "The fields are empty!!!!" from a MessageBox
    END DO
    ELSE
    DO
        TRY
        DO
            DECLARE clientName as string with value of
            D_ClientNameText.getText()

```


DECLARE issuerBank as string with value of
D_IssuerBankText.getText()

DECLARE bankAccount as string with value of
D_BankAccountText.getText()

DECLARE cardId as int with value of
D_CardIdText.getText()

DECLARE balanceAmount as int with value of
D_BalanceAmountText.getText()

DECLARE pinNumber as int with value of
D_PinNumberText.getText()

FOR BankCard Display_DebitCard from cards

DO

IF Display_DebitCard is an instance of DebitCard

DO

DOWNCAST Display_DebitCard to DebitCard
and store it to DebitCard_Object of DebitCard

DISPLAY DebitCard_Object

DISPLAY "Debit Card information are
displayed. Please check the display tab!!! "from
a MessageDialog

END DO

END DO

END DO

CATCH NumberFormatException nfe

DO

DISPLAY "Please provide required information only!!!"from a
MessageDialog

END DO

END DO

END DO

IF e.getSource() is D_WithdrawButton

DO

IF D_CardIDText.getText() is empty or
 D_WithdrawalAmountText.getText() is empty or
 D_WithdrawalDateLabel.getText() is empty or
 D_PinNumberrText.getText() is empty

DO

DISPLAY "The fields are empty!!!!" from a MessageDialog

END DO

ELSE

DO

TRY

DO

DECLARE dateOfWithdrawal as string with value of
 D_WithdrawalDateLabel.getText()

DECLARE cardID as int with value of
 D_CardIDText.getText()

DECLARE pinNumbeR as int with value of
 D_PinNumberrText.getText()

DECLARE withdrawalAmount as int with value of
 D_WithdrawalAmountText

FOR BankCard Withdraw_DebitCard from cards

DO

IF Withdraw_DebitCard is an instance of DebitCard

DO

DOWNCAST Withdraw_DebitCard to
DebitCard and store it to DebitCard_Object of
DebitCard

IF DebitCard_Object.getCardId() is equal to
cardID

DO

CALL method withdraw of
DebitCard_Object with parameters
withdrawalAmount, dateOfWithdrawal,
pinNumberR

DISPLAY "Your Debit Card has been
withdrawn successfully!!!" from a
MessageDialog

END DO

ELSE

DO

DISPLAY "Debit Card doesn't exist"
from a MessageDialog

END DO

END DO

END DO

END DO

CATCH NumberFormatException nfe

DO

DISPLAY "Please provide required information only!!!" from a
MessageDialog

END DO

END DO

END DO

IF e.getSource() is D_DisplayyButton

DO

IF D_CardIdText.getText() is empty or D_ClientNameText.getText() is empty or D_IssuerBankText.getText() is empty or D_BankAccountText.getText() is empty or D_BalanceAmountText.getText() is empty or D_PinNumberText.getText() is empty or D_CardIDText.getText() is empty or D_WithdrawalAmountText.getText() is empty or D_WithdrawalDateLabel.getText() is empty or D_PinNumberrText.getText() is empty

DO

DISPLAY "The fields are empty!!!!" from a MatDialog

END DO

ELSE

DO

TRY

DO

DECLARE clientName as string with value of D_ClientNameText.getText()

DECLARE issuerBank as string with value of D_IssuerBankText.getText()

DECLARE bankAccount as string with value of D_BankAccountText.getText()

DECLARE dateOfWithdrawal as string with value of D_WithdrawalDateLabel.getText()

DECLARE cardId as int with value of D_CardIdText.getText()

DECLARE balanceAmount as int with value of D_BalanceAmountText.getText()

DECLARE pinNumber as int with value of D_PinNumberText.getText()

DECLARE cardID as int with value of D_CardIDText.getText()

DECLARE pinNumbeR as int with value of D_PinNumberrText.getText()

DECLARE withdrawalAmount as int with value of
D-WithdrawalAmountText

FOR BankCard Displayy_DebitCard from cards

DO

IF Displayy_DebitCard is an instance of DebitCard

DO

DOWNCAST Displayy_DebitCard to DebitCard
and store it to DebitCard_Object of DebitCard

DISPLAY DebitCard_Object

DISPLAY " All Debit Card informations are
displayed. Please check the display tab!!! "from
a MatDialog

END DO

END DO

END DO

CATCH NumberFormatException nfe

DO

DISPLAY "Please provide required information only!!!"from a
MatDialog

END DO

END DO

END DO

IF e.getSource() is C_AddButton

DO

IF C_CardIdText.getText() is empty or C_ClientNameText.getText() is
empty or C_IssuerBankText.getText() is empty or
C_BankAccountText.getText() is empty or
C_BalanceAmountText.getText() is empty or
C_CVCNumberText.getText() is empty or C_InterestRateText.getText() is
empty or C_ExpirationDateLabel() is empty

```

DO
    DISPLAY "The fields are empty!!!!" from a MessageDialog
END DO
ELSE
DO
    TRY
    DO
        DECLARE clientName as string with value of
        C_ClientNameText.getText()

        DECLARE issuerBank as string with value of
        C_IssuerBankText.getText()

        DECLARE bankAccount as string with value of
        C_BankAccountText.getText()

        DECLARE expirationDate as string with value of
        C_ExpirationDateLabel()

        DECLARE cardId as int with value of
        C_CardIdText.getText()

        DECLARE balanceAmount as int with value of
        C_BalanceAmountText.getText()

        DECLARE cvcNumber as int with value of
        C_CVCNumberText.getText()

        DECLARE interestRate as double with value of
        C_InterestRateText.getText()

        DECLARE Existence_of_Card as boolean with the value of
        false

        FOR BankCard CreditCard from cards
        DO
            IF CreditCard is an instance of CreditCard

```

```

DO
    DOWNCAST CreditCard to CreditCard and
    store it to CreditCard_Object of CreditCard
    IF CreditCard_Object.getCardId() is equal to
    cardId
        DO
            DECLARE Existence_of_Card as true
        END DO
    END DO
END DO
IF Existence_of_Card is equal to false
    DO
        CREATE a new CreditCard object with parameters
        (balanceAmount, cardId, bankAccount, issuerBank,
        clientName, cvcNumber, interestRate, expirationDate)
        ADD CreditCard_Object to the end of the cards list
        DISPLAY "Your Credit Card has been added
        successfully!!!" from a MatDialog
    END DO
ELSE
    DO
        DISPLAY "Credit Card already exists!!!" from a
        MatDialog
    END DO
END DO
CATCH NumberFormatException nfe
    DO
        DISPLAY "Please provide required information only!!!" from
        a MatDialog
    END DO
END DO

```

END DO

IF e.getSource() is C_DisplayButton

DO

IF C_CardIdText.getText() is empty or C_ClientNameText.getText() is empty or C_IssuerBankText.getText() is empty or C_BankAccountText.getText() is empty or C_BalanceAmountText.getText() is empty or C_CVCNumberText.getText() is empty or C_InterestRateText.getText() is empty or C_ExpirationDateLabel() is empty

DO

DISPLAY "The fields are empty!!!!" from a MessageDialog

END DO

ELSE

DO

TRY

DO

DECLARE clientName as string with value of C_ClientNameText.getText()

DECLARE issuerBank as string with value of C_IssuerBankText.getText()

DECLARE bankAccount as string with value of C_BankAccountText.getText()

DECLARE expirationDate as string with value of C_ExpirationDateLabel()

DECLARE cardId as int with value of C_CardIdText.getText()

DECLARE balanceAmount as int with value of C_BalanceAmountText.getText()

DECLARE cvcNumber as int with value of C_CVCNumberText.getText()

DECLARE interestRate as double with value of
C_InterestRateText.getText()

FOR BankCard Display_CreditCard from cards

DO

IF Display_CreditCard is an instance of CreditCard

DO

DOWNCAST Display_CreditCard to
CreditCard and store it to CreditCard_Object of
CreditCard

DISPLAY CreditCard_Object

DISPLAY "Credit Card information are
displayed. Please check the display tab!!! "from
a MatDialog

END DO

END DO

END DO

CATCH NumberFormatException nfe

DO

DISPLAY "Please provide required information only!!!"from a
MatDialog

END DO

END DO

END DO

IF e.getSource() is C_SetButton

DO

IF C_CardIDText.getText() is empty or C_CreditLimitText.getText()is
empty or C_GracePeriodText.getText() is empty

DO

DISPLAY "The fields are empty!!!!" from a MatDialog

```

END DO
ELSE
DO
    TRY
    DO
        DECLARE cardID as int with value of
        C_CardIDText.getText()

        DECLARE gracePeriod as int with value of
        C_GracePeriodText.getText()

        DECLARE creditLimit as double with value of
        C_CreditLimitText.getText()

        FOR BankCard Set_CreditCard from cards
        DO
            IF Set_CreditCard is an instance of CreditCard
            DO
                DOWNCAST Set_CreditCard to CreditCard
                and store it to CreditCard_Object of CreditCard

                IF CreditCard_Object.getCardId() is equal to
                cardID
                DO
                    IF creditLimit is lesser and equal to
                    CreditCard_Object.getBalanceAmount()
                    multiplied by 2.5
                    DO
                        CALL method setCreditLimit of
                        CreditCard_Object with
                        parameters creditLimit,
                        gracePeriod
                    
```

DISPLAY "Your Credit Limit has
been set successfully!!!"from a
MessageDialog

END DO

END DO

ELSE

DO

DISPLAY "Credit Card doesn't exist."
from a MessageDialog

END DO

END DO

END DO

END DO

CATCH NumberFormatException nfe

DO

DISPLAY "Please provide required information only!!!" from
a MessageDialog

END DO

END DO

END DO

IF e.getSource() is C_DisplayyButton

DO

IF C_CardIdText.getText() is empty or C_ClientNameText.getText() is
empty or C_IssuerBankText.getText() is empty or
C_BankAccountText.getText() is empty or
C_BalanceAmountText.getText() is empty or
C_CVCNumberText.getText() is empty or C_InterestRateText.getText() is
empty or C_ExpirationDateLabel() is empty or C_CardIDText.getText() is
empty or C_CreditLimitText.getText() is empty or
C_GracePeriodText.getText() is empty

DO

```

        DISPLAY "The fields are empty!!!!" from a MatDialog
    END DO
ELSE
DO
    TRY
    DO
        DECLARE clientName as string with value of
        C_ClientNameText.getText()

        DECLARE issuerBank as string with value of
        C_IssuerBankText.getText()

        DECLARE bankAccount as string with value of
        C_BankAccountText.getText()

        DECLARE expirationDate as string with value of
        C_ExpirationDateLabel()

        DECLARE cardId as int with value of
        C_CardIdText.getText()

        DECLARE balanceAmount as int with value of
        C_BalanceAmountText.getText()

        DECLARE cvcNumber as int with value of
        C_CVCNumberText.getText()

        DECLARE cardID as int with value of
        C_CardIDText.getText()

        DECLARE gracePeriod as int with value of
        C_GracePeriodText.getText()

        DECLARE creditLimit as double with value of
        C_CreditLimitText.getText()

        DECLARE interestRate as double with value of
        C_InterestRateText.getText()

        FOR BankCard Displayy_CreditCard from cards

```

```

DO
    IF Displayy_CreditCard is an instance of CreditCard
        DO
            DOWNCAST Displayy_CreditCard to
            CreditCard and store it to CreditCard_Object of
            CreditCard

            DISPLAY CreditCard_Object

            DISPLAY "Credit Card information are
            displayed. Please check the display tab!!! "from
            a MatDialog

        END DO
    END DO
END DO

CATCH NumberFormatException nfe
DO
    DISPLAY "Please provide required information only!!!"from a
    MatDialog
END DO
END DO
END DO

IF e.getSource() is C_CancelButton
DO
    IF C_CarDIDText.getText() is empty
        DO
            DISPLAY "The fields are empty!!!!" from a MatDialog
        END DO
    ELSE
        DO
            TRY

```

DO

DECLARE carDID as int with value of
C_CarDIDText.getText()

FOR BankCard Cancel_CreditCard from cards

DO

IF Cancel_CreditCard is an instance of CreditCard

DO

DOWNCAST Cancel_CreditCard to CreditCard
and store it to CreditCard_Object of CreditCard

IF CreditCard_Object.getCardId() is equal to
carDID

DO

CALL method cancelCreditCard of
CreditCard_Object with no parameters

REMOVE Cancel_CreditCard from
ArrayList cards

DISPLAY "Credit Card has been
cancelled successfully!! "from a
AlertDialog

END DO

ELSE

DO

DISPLAY "Credit Card hasn't been
cancelled" from a AlertDialog

END DO

END DO

END DO

END DO

CATCH NumberFormatException nfe

DO

DISPLAY "Please provide required information only!!!" from
a MessageDialog

END DO

END DO

END DO

IF e.getSource() is C_DisplayyyButton

DO

IF C_CardIdText.getText() is empty or C_ClientNameText.getText() is
empty or C_IssuerBankText.getText() is empty or
C_BankAccountText.getText() is empty or
C_BalanceAmountText.getText() is empty or
C_CVCNumberText.getText() is empty or C_InterestRateText.getText() is
empty or C_ExpirationDateLabel() is empty or C_CardIDText.getText() is
empty or C_CreditLimitText.getText() is empty or
C_GracePeriodText.getText() is empty or C_CarDIDText.getText() is
empty

DO

DISPLAY "The fields are empty!!!!" from a MessageDialog

END DO

ELSE

DO

TRY

DO

DECLARE clientName as string with value of
C_ClientNameText.getText()

DECLARE issuerBank as string with value of
C_IssuerBankText.getText()

DECLARE bankAccount as string with value of
C_BankAccountText.getText()

DECLARE expirationDate as string with value of
C_ExpirationDateLabel()

DECLARE cardId as int with value of
C_CardIdText.getText()

DECLARE balanceAmount as int with value of
C_BalanceAmountText.getText()

DECLARE cvcNumber as int with value of
C_CVCNumberText.getText()

DECLARE cardID as int with value of
C_CardIDText.getText()

DECLARE gracePeriod as int with value of
C_GracePeriodText.getText()

DECLARE carDID as int with value of
C_CarDIDText.getText()

DECLARE creditLimit as double with value of
C_CreditLimitText.getText()

DECLARE interestRate as double with value of
C_InterestRateText.getText()

FOR BankCard Displayyy_CreditCard from cards

DO

IF Displayyy_CreditCard is an instance of CreditCard

DO

DOWNCAST Displayyy_CreditCard to
CreditCard and store it to CreditCard_Object of
CreditCard

DISPLAY CreditCard_Object

DISPLAY "All Credit Card information are
displayed. Please check the display tab!!! "from
a MatDialog

END DO

END DO

END DO


```

CATCH NumberFormatException nfe
DO
    DISPLAY "Please provide required information only!!!"from a
    MessageDialog
END DO
END DO
END DO

IF e.getSource() is C_DisplayyyButton
DO
    SET D_CardIdText to ""
    SET D_ClientNameText to ""
    SET D_IssuerBankText to ""
    SET D_BankAccountText to ""
    SET D_BalanceAmountText to ""
    SET D_PinNumberText to ""

    SET D_CardIDText to ""
    SET D-WithdrawalAmountText to ""
    SET D_PinNumberrText to ""

    SET C_CardIdText to ""
    SET C_ClientNameText to ""
    SET C_IssuerBankText to ""
    SET C_BankAccountText to ""
    SET C_BalanceAmountText to ""
    SET C_CVCNumberText to ""
    SET C_InterestRateText to ""

```

SET C_CardIDText to ""

SET C_GracePeriodText to ""

SET C_CreditLimitText to ""

SET C_CarDIDText to ""

END DO

END DO

CREATE main method

DO

CREATE new object of the class BankGUI and assign it to variable obj

END DO

4. Method descriptions

A method is a block or set of code or collection that is grouped together in order to perform certain task and operations. It helps to make code more reusable. Here, we have used the accessor and mutator methods, which are instance methods (Anon., n.d.).

i. Method description of BankGUI

Methods	Description
BankGUI()	BankGUI is a method that is called at the start of the BankGUI class. The method is used to enter data into the GUI's frame. The method's return type is void. It does not accept any parameters.
actionPerformed (ActionEvent e)	actionPerformed is a method defined by the ActionListener interface. When an event occurs, the method executes any task, and the event is listened to with actionPerformed. The method's return type is void. It associates an object with the parameter(ActionEvent).
D_AddButton	This button inserts the Debit Card object created by the user's data into the Debit Card's add frame. The object is generated using the data entered by the user as a parameter. After that, the object is added to the array list.
D_WithdrawButton	This button takes the value entered into the Debit Card withdraw and runs the withdraw method of the Debit Card class with the parameters entered by the user.
C_AddButton	This button inserts the Credit Card object created by the user's data into the Credit Card's add frame. The object is generated using the data entered by the user as a parameter. After that, the object is added to the array list.

C_SetButton	This button takes the value entered into the Credit Card set frame and runs the setCreditLimit method of the Credit Card class with the parameters entered by the user.
C_CancelButton	This button takes the value entered into the Credit Card cancel frame and runs the cancelCreditCard method of the Credit Card class with the parameters entered by the user.
display	The value of the Debit Card or Credit Card objects created by invoking the display method in the Debit Card or Credit Card Class is displayed by this button.
clear	This button removes the fields in the GUI that the user has written over. It clears the text boxes and makes them empty again.
public static void main()	This method from BankGUI is used to access the object. There is no return type for this procedure. We can also run the GUI directly using the primary method.

Table 1 BankGUI Method

5. Inspection

Inspection of code is the process of verification which assesses the accuracy of the code contained within the object. Code inspections are effective at identifying faults in functionality, dependability, and maintainability of the program (McGovern, 2003).

The following tests are carried out to examine the compatibility of the program to use in the actual world.

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.

Test No.	1
Objective:	To Test that the program can be compiled and run using the command prompt.
Action:	<ul style="list-style-type: none"> ➤ Open command prompt ➤ Type ls ➤ Type java BankGUI.java ➤ The program is opened.
Expected Result:	The program would be compiled and run using the command prompt
Actual Result:	The program was compiled and run through the command prompt.
Conclusion:	The test is successful.

Table 2 Test that the program can be compiled and run using the command prompt

Output:

```

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Rashi\Desktop\22067683_Rashi Maharjan> ls

Directory: C:\Users\Rashi\Desktop\22067683_Rashi Maharjan

Mode                LastWriteTime         Length Name
----                -
-a-----         2/8/2023   9:45 AM             2218 BankCard.class
-a-----         2/8/2023   9:45 AM             885 BankCard.ctxt
-a-----         2/8/2023   9:45 AM            1840 BankCard.java
-a-----         5/3/2023  11:21 AM           17397 BankGUI.class
-a-----         5/3/2023  11:21 AM             263 BankGUI.ctxt
-a-----         5/3/2023  11:23 AM           37525 BankGUI.java
-a-----         4/10/2023   9:34 PM            2808 CreditCard.class
-a-----         4/10/2023   9:34 PM            1034 CreditCard.ctxt
-a-----         2/8/2023   9:40 AM            2370 CreditCard.java
-a-----         4/29/2023   7:30 AM            2557 DebitCard.class
-a-----         4/29/2023   7:30 AM             912 DebitCard.ctxt
-a-----         4/29/2023   7:27 AM            2364 DebitCard.java
-a-----         5/2/2023  11:03 AM            1414 package.bluej
-a-----         1/5/2023   7:14 AM             480 README.TXT

PS C:\Users\Rashi\Desktop\22067683_Rashi Maharjan> java BankGUI.java
Note: BankGUI.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

```

Figure 6 Screenshot of class being compiled

Banking System

Debit Card		Credit Card	
Card Id: <input type="text"/>	Balance Amount: <input type="text"/>	Card Id: <input type="text"/>	Balance Amount: <input type="text"/>
Client Name: <input type="text"/>	Pin Number: <input type="text"/>	Client Name: <input type="text"/>	CVC Number: <input type="text"/>
Issuer Bank: <input type="text"/>		Issuer Bank: <input type="text"/>	Interest Rate: <input type="text"/>
Bank Account: <input type="text"/>		Bank Account: <input type="text"/>	
		Expiration Date: <input type="text"/> 1 <input type="text"/> January <input type="text"/> 2011 <input type="text"/>	
<input type="button" value="Display"/> <input type="button" value="Add to Debit Card"/>		<input type="button" value="Display"/> <input type="button" value="Add to Credit Card"/>	
Card Id: <input type="text"/>		Card Id: <input type="text"/>	
Withdrawal Amount: <input type="text"/>		Credit Limit: <input type="text"/>	
Withdrawal Date: <input type="text"/> 1 <input type="text"/> January <input type="text"/> 2011 <input type="text"/>		Grace Period: <input type="text"/>	
Pin Number: <input type="text"/>			
<input type="button" value="Display"/> <input type="button" value="Withdraw"/>		<input type="button" value="Display"/> <input type="button" value="Set Credit Limit"/>	
		Card Id: <input type="text"/>	
		<input type="button" value="Display"/> <input type="button" value="Cancel Credit Card"/>	
<input type="button" value="Clear"/>			

Figure 7 Screenshot of BankGUI running

Test 2: Evidences should be shown of: a. Add DebitCard b. Add CreditCard c. Withdraw amount from Debit card d. Set the credit limit e. Remove the credit card

Test No.	2a
Objective:	To show evidences of adding Debit Card
Action:	<ul style="list-style-type: none"> ➤ BankGUI is compiled and following input is given: Card Id: 1 Client Name: Raj Issuer Bank: Nabil Bank Bank Account: aa12 Balance Amount: 12550 Pin Number: 12345 ➤ “Add to Debit Card” Button is pressed. ➤ “Display” Button is pressed to inspect addition of Debit Card to the ArrayList of the BankGUI class.
Expected Result:	The adding of Debit Card would be granted.
Actual Result:	The adding of Debit Card was done.
Conclusion:	The test is successful.

Table 3 Adding Debit Card

Output:

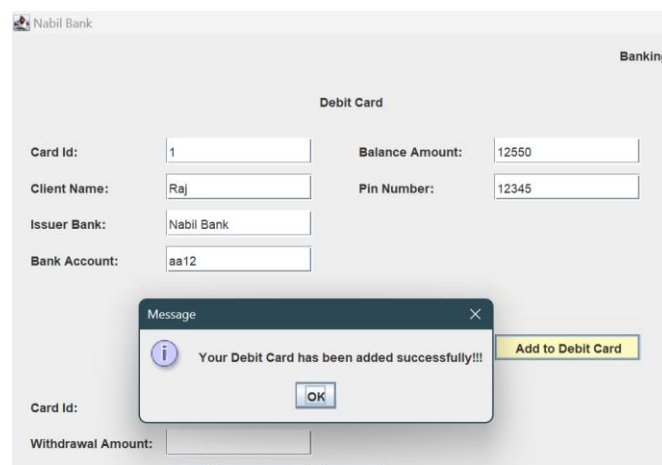


Figure 8 Screenshot of Adding Debit Card

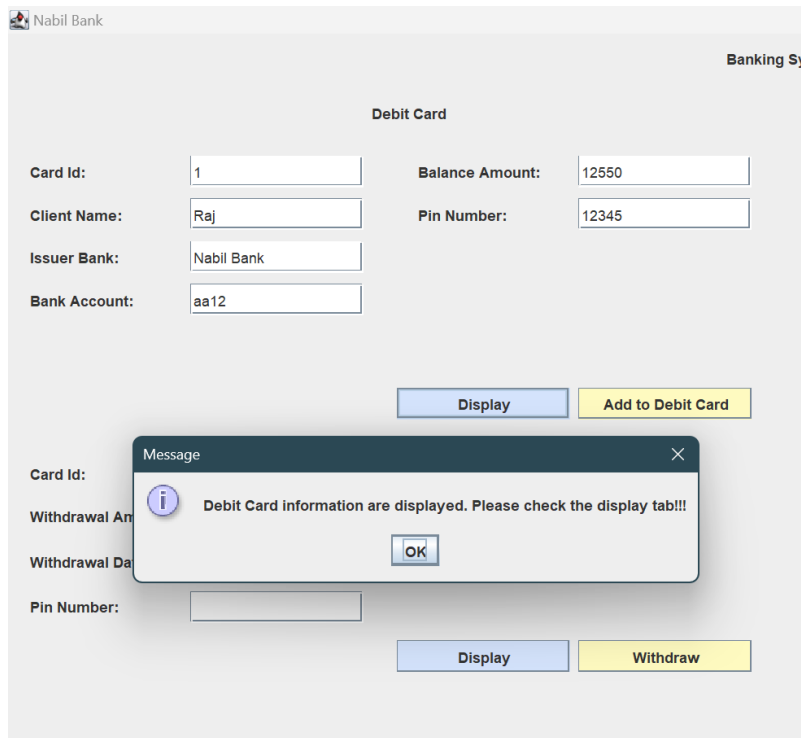


Figure 9 Screenshot of clicking Display Button when adding Debit Card

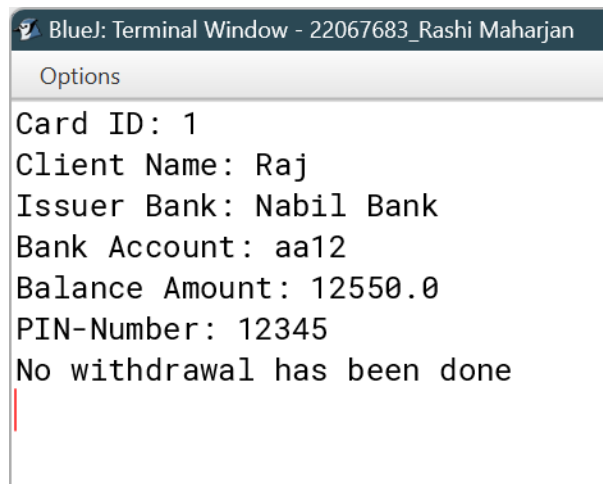


Figure 10 Screenshot of information when Debit Card is added.

Test No.	2b
Objective:	To show evidences of adding Credit Card
Action:	<ul style="list-style-type: none"> ➤ BankGUI is compiled and following input is given: Card Id: 1 Client Name: Raj Issuer Bank: Nabil Bank Bank Account: aa12 Expiration Date: 30 March 2025 Balance Amount: 12550 CVC Number: 123 Interest Rate: 5 ➤ Add to Credit Card Button is pressed. ➤ “Display” Button is pressed to inspect addition of Credit Card to the ArrayList of the BankGUI class.
Expected Result:	The adding of Credit Card would be granted.
Actual Result:	The adding of Credit Card was done.
Conclusion:	The test is successful.

Table 4 Adding of Credit Card

Output:

The screenshot shows a Java Swing window titled "System" with a "Credit Card" form. The form contains the following fields and values:

- Card Id: 1
- Client Name: Raj
- Issuer Bank: Nabil Bank
- Bank Account: aa12
- Expiration Date: 30 March 2025
- Balance Amount: 12550
- CVC Number: 123
- Interest Rate: 5

An "Add to Credit Card" button is located at the bottom right of the form. A message dialog box is overlaid on the form, displaying the message: "Your Credit Card has been added successfully!!!" with an "OK" button.

Figure 11 Screenshot of Adding Credit Card

The screenshot shows a 'Banking System' window with a 'Credit Card' section. The form contains the following fields and values:

Field	Value
Card Id:	1
Client Name:	Raj
Issuer Bank:	Nabil Bank
Bank Account:	aa12
Expiration Date:	30 March 2025
Balance Amount:	12550
CVC Number:	123
Interest Rate:	5

Buttons visible include 'Add to Debit Card', 'Display', 'Add to Credit Card', 'Withdraw', 'Set Credit Limit', 'Clear', and 'Cancel Credit Card'. A 'Message' dialog box is displayed in the center with the text: 'Credit Card information are displayed. Please check the display tab!!!' and an 'OK' button.

Figure 12 Screenshot of clicking Display Button when adding Credit Card

The screenshot shows a BlueJ Terminal Window titled 'BlueJ: Terminal Window - 22067683_Rashi Maharjan'. The output text is as follows:

```
Options
Card ID: 1
Client Name: Raj
Issuer Bank: Nabil Bank
Bank Account: aa12
Balance Amount: 12550.0
No credit has been granted yet.
```

Figure 13 Screenshot of information when Credit Card is added.

Test No.	2c
Objective:	To show evidences of withdraw amount from Debit Card
Action:	<p>➤ BankGUI is compiled and following input is given:</p> <p>Card Id: 1</p> <p>Client Name: Raj</p> <p>Issuer Bank: Nabil Bank</p> <p>Bank Account: aa12</p> <p>Balance Amount: 12550</p> <p>Pin Number: 12345</p> <p>Card Id: 1</p> <p>Withdrawal Amount: 550</p> <p>Withdrawal Date: 5 May 2023</p> <p>Pin Number: 12345</p> <p>➤ Withdraw Button is pressed.</p> <p>➤ “Display” Button is pressed to inspect withdraw of Debit Card to the ArrayList of the BankGUI class.</p>
Expected Result:	The withdrawing amount from Debit Card would be granted.
Actual Result:	The withdraw amount from Debit Card was done.
Conclusion:	The test is successful.

Table 5 Withdraw amount from Credit Card

Output:

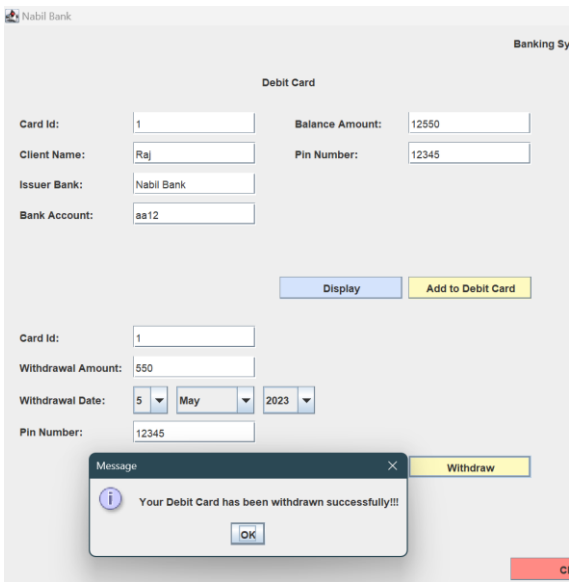


Figure 14 Screenshot of withdrawing amount from Debit Card

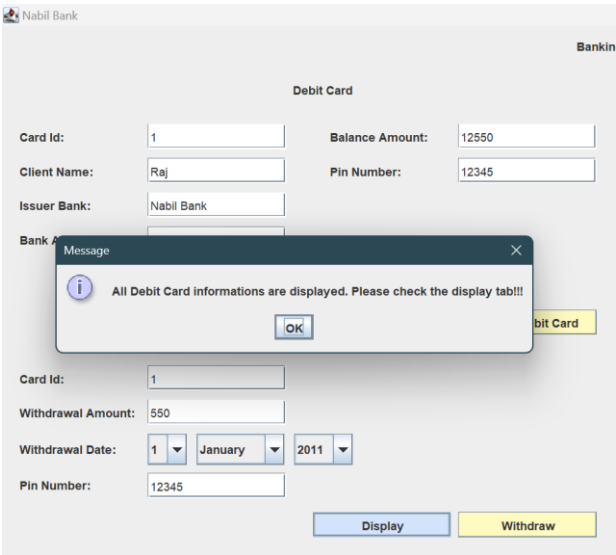


Figure 15 Screenshot of clicking Display Button when withdrawing Debit Card

```
BlueJ: Terminal Window - 22067683_Rashi Maharjan
Options
New Balance: 12000.0
Card ID: 1
Client Name: Raj
Issuer Bank: Nabil Bank
Bank Account: aa12
Balance Amount: 12000.0
PIN-Number: 12345
Withdrawal-Amount: 550
```

Figure 16 Screenshot of information when Debit Card is withdrawn

Test No.	2d
Objective:	To show evidences of setting Credit Limit
Action:	<p>➤ BankGUI is compiled and following input is given:</p> <p>Card Id: 1</p> <p>Client Name: Raj</p> <p>Issuer Bank: Nabil Bank</p> <p>Bank Account: aa12</p> <p>Expiration Date: 30 March 2025</p> <p>Balance Amount: 12550</p> <p>CVC Number: 123</p> <p>Interest Rate: 5</p> <p>Card Id: 1</p> <p>Credit Limit: 3</p> <p>Grace Period: 4</p> <p>➤ Set Credit Limit Button is pressed.</p>
Expected Result:	The setting of Credit Limit would be granted.
Actual Result:	The setting Credit Limit was done.
Conclusion:	The test is successful.

Table 6 Setting the Credit Limit

Output:

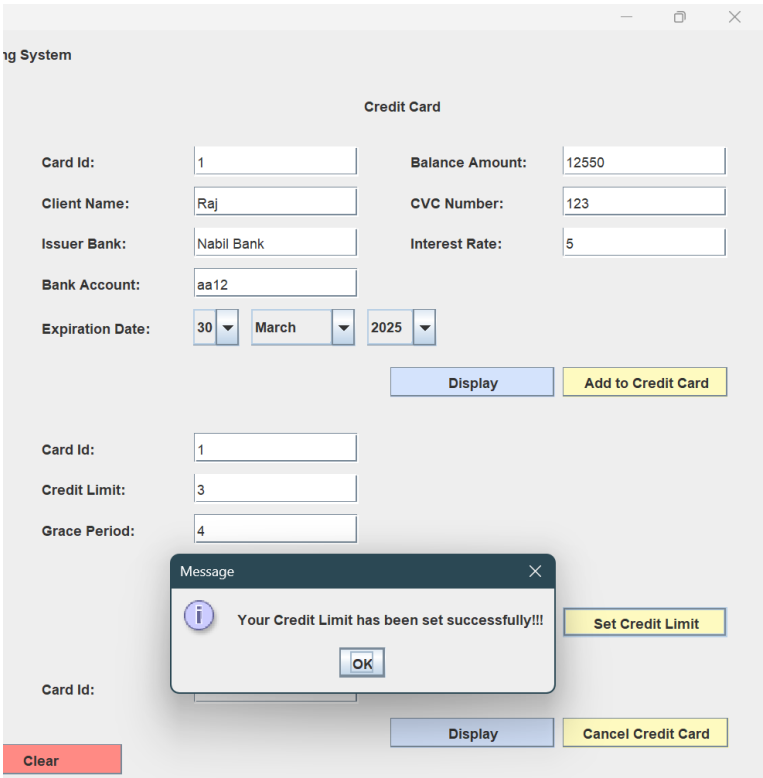


Figure 17 Screenshot of Setting the Credit Limit

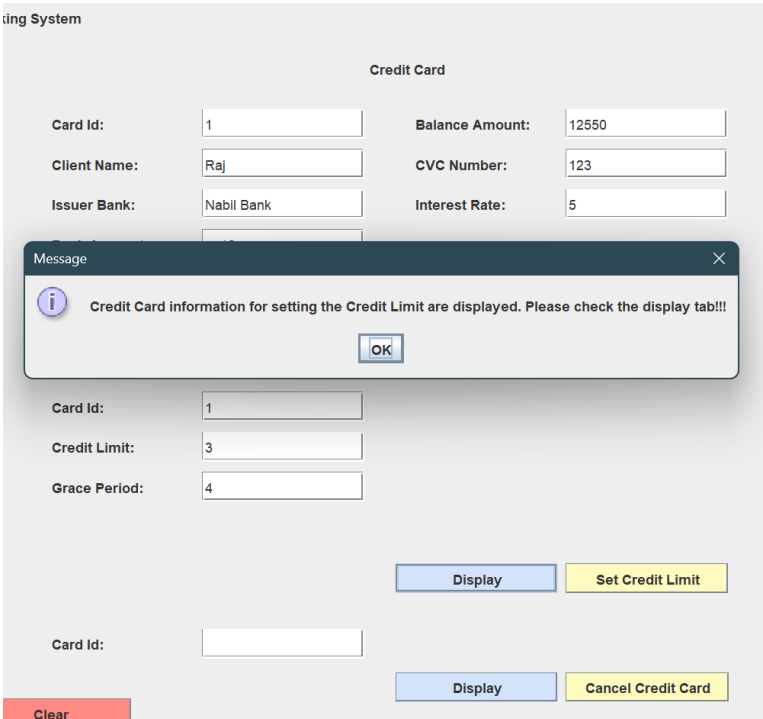
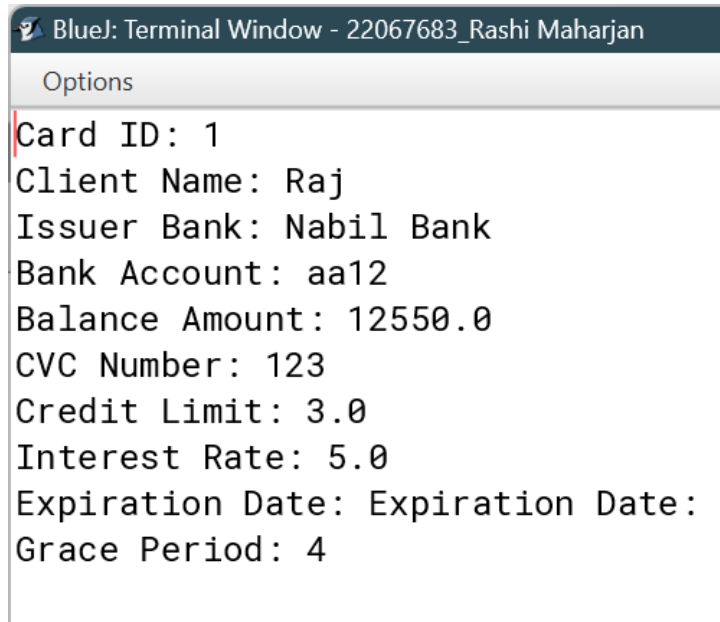


Figure 18 Screenshot of clicking Display Button when setting the credit limit

A screenshot of a BlueJ terminal window. The title bar reads "BlueJ: Terminal Window - 22067683_Rashi Maharjan". Below the title bar is a tab labeled "Options". The terminal area displays the following text:

```
Card ID: 1
Client Name: Raj
Issuer Bank: Nabil Bank
Bank Account: aa12
Balance Amount: 12550.0
CVC Number: 123
Credit Limit: 3.0
Interest Rate: 5.0
Expiration Date: Expiration Date:
Grace Period: 4
```

Figure 19 Screenshot of information when Credit Limit is set

Test No.	2e
Objective:	To show evidences of removing Credit Card
Action:	<p>➤ BankGUI is compiled and following input is given:</p> <p>Card Id: 1</p> <p>Client Name: Raj</p> <p>Issuer Bank: Nabil Bank</p> <p>Bank Account: aa12</p> <p>Expiration Date: 30 March 2025</p> <p>Balance Amount: 12550</p> <p>CVC Number: 123</p> <p>Interest Rate: 5</p> <p>Card Id: 1</p> <p>Credit Limit: 3</p> <p>Grace Period: 4</p> <p>Card Id: 1</p> <p>➤ Cancel Credit Card Button is pressed.</p>
Expected Result:	The removing of Credit Card would be granted.
Actual Result:	The removing of Credit Card was done.
Conclusion:	The test is successful.

Table 7 Remove the Credit Card

Output:

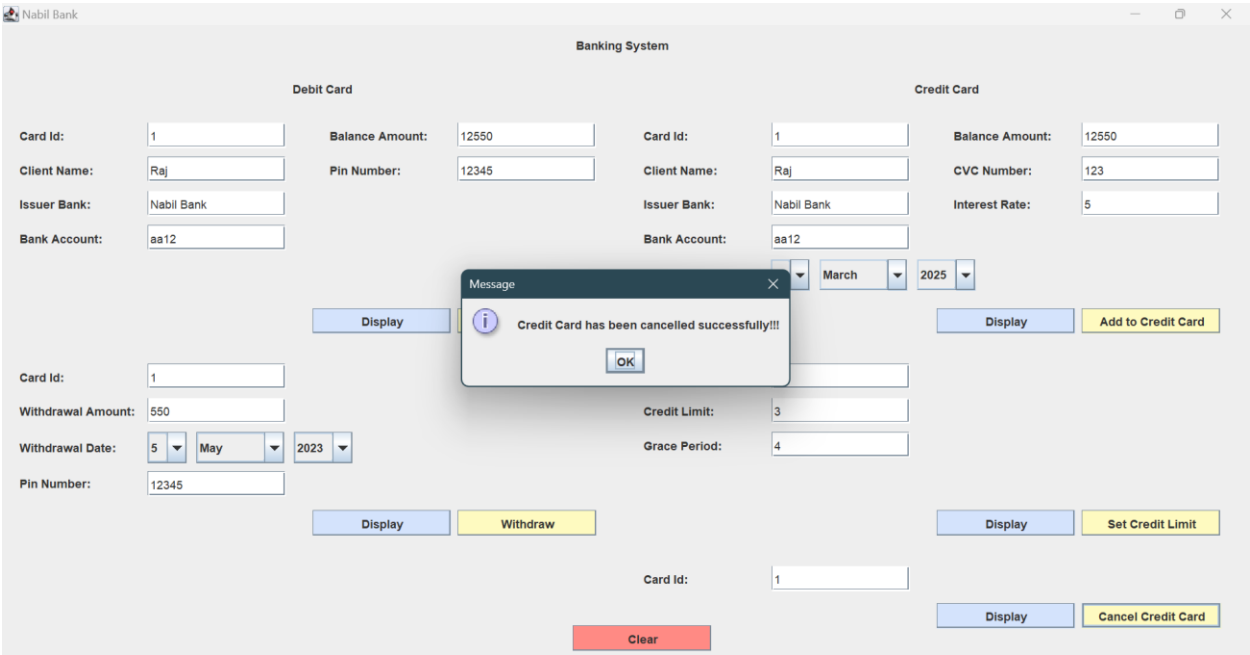


Figure 20 Screenshot of Removing the Credit Card

Test 3: Test that appropriate dialog boxes appear when unsuitable values are entered for the Card ID, (include a screenshot of the dialog box, together with a corresponding screenshot of the GUI, showing the values that were entered).

Test No.	3
Objective:	To test that appropriate dialog boxes appear when unsuitable values are entered for the Card ID.
Action:	<ul style="list-style-type: none"> ➤ BankGUI is compiled and following input is given: Card Id: aa Client Name: Raj Issuer Bank: Nabil Bank Bank Account: aa12 Balance Amount: 12550 Pin Number: 12345 ➤ Add to Debit Card Button is pressed.
Expected Result:	The dialog boxes would be appeared when unsuitable values are entered.
Actual Result:	The dialog boxed was appeared.
Conclusion:	The test is successful.

Table 8 Testing whether the dialog box appears when unsuitable values are entered.

Output:

The screenshot displays a banking application window titled 'Nabil Bank'. The main form is titled 'Debit Card' and contains several input fields and buttons. A message dialog box is overlaid on the form, displaying the text 'Please provide required information only!!!' with an 'OK' button. The form fields include:

- Card Id:** aa
- Balance Amount:** 12550
- Client Name:** Raj
- Pin Number:** 12345
- Issuer Bank:** Nabil Bank
- Bank Account:** aa12

Below the message dialog box, there are additional fields and buttons:

- Card Id:** (empty field)
- Withdrawal Amount:** (empty field)
- Withdrawal Date:** 1 January 2011 (using dropdown menus)
- Pin Number:** (empty field)
- Buttons:** 'Add to Debit Card' (yellow), 'Display' (blue), and 'Withdraw' (yellow).

Figure 21 Screenshot of the dialog box which appeared when unsuitable values were entered

6. Error detection and Correction

Error detection is the method of detecting errors and faults in the program where as the process of resolving that error is called error correction.

The following are the errors that were detected while performing the programming and its respective solution;

Error 1: Syntax Error

A syntax error occurs when a program's source code has an error. This kind of error is a minor grammatical error that affects only one character. For example, a missing semi-colon at the end of a line, an extra bracket at the end of a function or a spelling mistake can result in a syntax error (George, 2022).



```

55 //create components
56 Label TitleText = new JLabel("Banking System");
57

```

Figure 22 Syntax Error

In the above figure, an error has been detected while creating components because the syntax to add label is incorrect.



```

55 //create components
56 JLabel TitleText = new JLabel("Banking System");
57

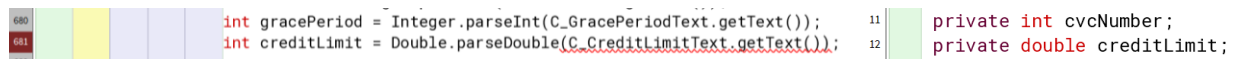
```

Figure 23 Syntax Error Solution

The error was corrected by rewriting the correct the syntax for creating label as J in JLabel was missing.

Error 2: Semantic Error

The semantic error can be caused by using the incorrect variable, operator, or order of operations. A semantic mistake relates to meaning (Anon., n.d.).



```

600 int gracePeriod = Integer.parseInt(C_GracePeriodText.getText());
601 int creditLimit = Double.parseDouble(C_CreditLimitText.getText());
11 private int cvcNumber;
12 private double creditLimit;

```

Figure 24 Semantic Error

An error has been located in the BankGUI class as the data type for the attribute “creditLimit” is assigned as integer but as double in Credit Card class.

```
682 double creditLimit = Double.parseDouble(C_CreditLimitText.getText());
683
```

Figure 25 Sematic Error Solution

The error has been corrected by changing the data type into double in BankGUI class.

Error 3: Logical Error

A logical error in Java is a type of error when a program compiles and runs without any errors but fails to yield any results (Anon., 2019).

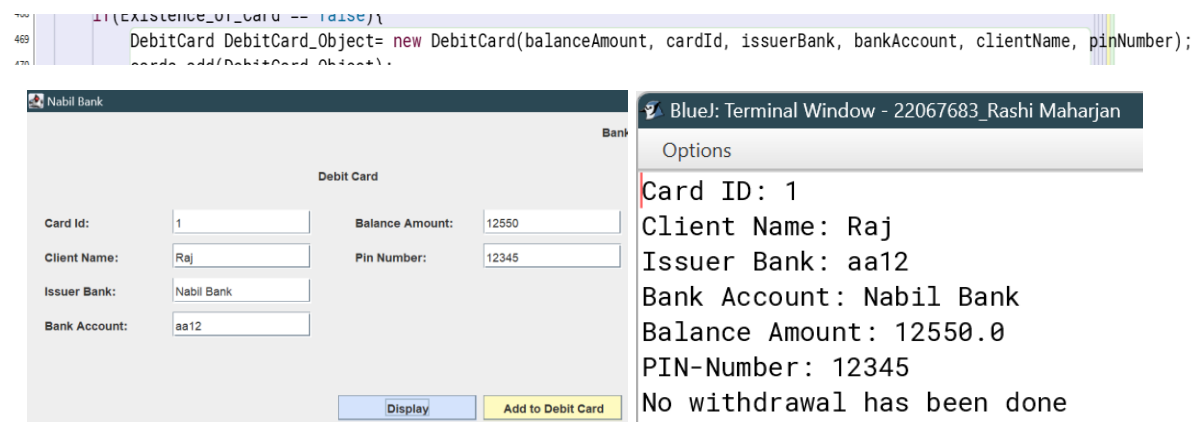


Figure 26 Logical Error

In the above figure, the value of Issuer Bank and Bank Account is swapped. There wasn't any error displayed while compiling the program but the output was not as expected.

```

469 DebitCard DebitCard_Object= new DebitCard(balanceAmount, cardId, bankAccount, issuerBank, clientName, pinNumber);
470 cards.add(DebitCard_Object);

```

The screenshot shows a Java Swing application window titled "Nabil Bank". Inside the window, there is a "Debit Card" form with the following fields and values:

Field	Value
Card Id:	1
Client Name:	Raj
Issuer Bank:	Nabil Bank
Bank Account:	aa12
Balance Amount:	12550
Pin Number:	12345

At the bottom of the form, there are two buttons: "Display" (blue) and "Add to Debit Card" (yellow).

To the right of the form is a terminal window titled "BlueJ: Terminal Window - 22067683_Rashi Maharjan". It shows the following output:

```

Options
Card ID: 1
Client Name: Raj
Issuer Bank: Nabil Bank
Bank Account: aa12
Balance Amount: 12550.0
PIN-Number: 12345
No withdrawal has been done

```

Figure 27 Logical Error solution

The following error was resolved by changing the position of bankAccount and issuerBank in the code. As both are String data types, swapping their position didn't create any syntax error but their required values changed while displaying the output.

7. Conclusion

In conclusion, this java project describes how to use JAVA programming to create a Graphical User Interface (GUI) for a banking system including the components of debit card, and credit card. The array list of BankCard class is extracted and kept in BankGUI class. The code for this java project was originally written and compiled in BlueJ. In order to simulate GUI functionality, the project uses several Java programming techniques, including classes, objects, constructors, methods, looping statements, conditional statements, inheritance, and encapsulation. The project also demonstrates how the simulation was designed and implemented using the Object-Oriented Programming paradigm.

In order to create an effective software, a variety of principles like code writing, pseudo code, testing, and class diagrams must be implemented which we got to learn while doing this coursework. This project taught me the value of modularity and code organization, which was one of the major lessons I got to learn.

Testing and problem fixing were following two crucial project components. We needed to thoroughly test our code to make sure it was operating as intended and to address any errors that we might run across. The hardest part of my coursework was to do coding and implement the coding instructions as this is a new concept to us that gave me the most insight into the subject module. The overall experience has been challenging and rewarding experience.

8. References

- Anon., 2019. *Types of Errors in Java*. [Online]
Available at: <https://www.gangboard.com/blog/types-of-errors-in-java/>
[Accessed 26 01 2023].
- Anon., n.d. [Online]
Available at: <https://www.javatpoint.com/method-in-java>
[Accessed 26 01 2023].
- Anon., n.d. *Semantic Error*. [Online]
Available at: <https://www.javatpoint.com/semantic-error>
[Accessed 25 01 2023].
- George, E., 2022. *What is a Syntax Error? How To Fix It*. [Online]
Available at: <https://clouddevelop.org/syntax-error/>
[Accessed 26 01 2023].
- Hagan, D. & Markham, S., 2000. *Teaching Java with the BlueJ Environment*, s.l.: s.n.
- Karatrantou, A. & Panagiotakopoulos, C., 2008. *Algorithm, Pseudo-Code and Lego Mindstorms Programming*, Venice(Italy): University of Patras.
- Liang, Y. D., 2012. *Introduction to Java Programming*. 9th ed. s.l.:Pearson Education.
- McGovern, J., 2003. *Java Web Services Architecture*. [Online]
Available at: <https://www.sciencedirect.com/topics/computer-science/code-inspection>
[Accessed 26 01 2023].
- Walker, A. & Matthew , M., 2022. *Guru99*. [Online]
Available at: <https://www.guru99.com/>
[Accessed 21 1 2022].

9. Appendix

i. Code of Bank Card

```
public class BankCard
{
    //attributes

    private int cardId;

    private String clientName;

    private String issuerBank;

    private String bankAccount;

    private double balanceAmount;


    //constructor

    public BankCard (double balanceAmount, int cardId, String bankAccount, String
issuerBank)
    {
        //attribute = parameter

        this.balanceAmount = balanceAmount;

        this.cardId = cardId;

        this.bankAccount = bankAccount;

        this.issuerBank = issuerBank;

        this.clientName = "";                                //empty string
    }
}
```

```
//mutator

public void setClientName(String clientName) {

    this.clientName = clientName;

}

public void setBalanceAmount(double balanceAmount) {

    this.balanceAmount = balanceAmount;

}

//accessor methods

public int getCardId() {

    return this.cardId;

}

public String getClientName() {

    return this.clientName;

}

public String getIssuerBank() {

    return this.issuerBank;

}

public String getBankAccount() {
```

```
        return this.bankAccount;
    }

    public double getBalanceAmount() {
        return this.balanceAmount;
    }

    //display method: void because it didn't return anything
    public void display() {
        System.out.println ("Card ID: "+cardId);
        If (this.clientName == "") {
            System.out.println ("Client name is not assigned");
        }
        else {
            System.out.println ("Client Name: "+ clientName);
        }
        System.out.println ("Issuer Bank: "+ issuerBank);
        System.out.println ("Bank Account: "+ bankAccount);
        System.out.println ("Balance Amount: "+ balanceAmount);
    }
}
```

ii. Code of Debit Card

```
public class DebitCard extends BankCard
{
    //attributes

    private int pinNumber;

    private int withdrawalAmount;

    private String dateOfWithdrawal;

    private boolean hasWithdrawn;


    //constructor

    public DebitCard (double balanceAmount, int cardId, String bankAccount, String
issuerBank, String clientName, int pinNumber) {

        super (balanceAmount, cardId, bankAccount, issuerBank); //attributes called
        from super class

        setClientName (clientName); // method called from super class

        this.pinNumber = pinNumber;

        this.hasWithdrawn = false;

    }


    //mutator

    public void setWithdrawalAmount (int withdrawalAmount) {

        this.withdrawalAmount = withdrawalAmount;

    }
```

```
//accessor methods

public int getPinNumber() {

    return this.pinNumber;

}

public int getWithdrawalAmount() {

    return this.withdrawalAmount;

}

public String getDateOfWithdrawal() {

    return this.dateOfWithdrawal;

}

public boolean getHasWithdrawn() {

    return this.hasWithdrawn;

}

//withdraw method

public void withdraw (int withdrawalAmount, String dateOfWithdrawal, int
pinNumber) {

    if (this.pinNumber == pinNumber && withdrawalAmount <=
super.getBalanceAmount()) {
```

```
        super.setBalanceAmount(super.getBalanceAmount() - withdrawalAmount);

        this.withdrawalAmount = withdrawalAmount;

        this.dateOfWithdrawal = dateOfWithdrawal;

        this.hasWithdrawn = true;

        System.out.println("New Balance: " + super.getBalanceAmount());

    }

    else if (this.pinNumber != pinNumber) {

        System.out.println ("The Entered PIN-Number is INVALID");

    }

    else {

        System.out.println ("The Balance is Insufficient");

    }

}

//display method

public void display() {

    super.display();

    System.out.println ("PIN-Number: " + pinNumber);

    if(hasWithdrawn == true) {

        System.out.println ("Withdrawal-Amount: " + withdrawalAmount);

        System.out.println ("Date-Of-WithDrawal: " + dateOfWithdrawal);

    }

    else{
```

```

        System.out.println ("No withdrawal has been done");
    }
}
}

```

iii. Code of Credit Card

```

public class CreditCard extends BankCard
{
    //attributes

    private int cvcNumber;

    private double creditLimit;

    private double interestRate;

    private String expirationDate;

    private int gracePeriod;

    private boolean isGranted;


    //constructor

    public CreditCard (double balanceAmount, int cardId, String bankAccount, String
issuerBank, String clientName, int cvcNumber, double interestRate, String
expirationDate) {

        super (balanceAmount, cardId, bankAccount, issuerBank);

        setClientName (clientName);

        this.cvcNumber = cvcNumber;
    }
}

```

```
this.interestRate = interestRate;

this.expirationDate = expirationDate;

this.isGranted = false;
}

//credit limit setter method

public void setCreditLimit (double creditLimit, int gracePeriod) {

    if (creditLimit <= 2.5 * super.getBalanceAmount()){

        this.creditLimit = creditLimit;

        this.gracePeriod = gracePeriod;

        this.isGranted = true;

    }else{

        System.out.println ("Credit cannot be issued.");

    }

}

//accessor method

public int getCvcNumber() {

    return this.cvcNumber;

}

public double getCreditLimit() {

    return this.creditLimit;

}
```



```
public double getInterestRate() {  
    return this.interestRate;  
}  
  
public String getExpirationDate() {  
    return this.expirationDate;  
}  
  
public int getGracePeriod() {  
    return this.gracePeriod;  
}  
  
public boolean getIsGranted() {  
    return this.isGranted;  
}  
  
  
//cancel card credit method  
  
public void cancelCreditCard() {  
    if (isGranted == true) {  
        cvcNumber = 0;  
        creditLimit = 0;  
        gracePeriod = 0;  
        isGranted = false;  
    }  
    else{  
        System.out.println ("No credit card to cancel.");  
    }  
}
```

```
}  
  
}  
  
//display method  
public void display(){  
    super.display();  
    if (isGranted == true) {  
        System.out.println ("CVC Number: " + cvcNumber);  
        System.out.println ("Credit Limit: " + creditLimit);  
        System.out.println ("Interest Rate: " + interestRate);  
        System.out.println ("Expiration Date: " + expirationDate);  
        System.out.println ("Grace Period: " + gracePeriod);  
    }  
    else{  
        System.out.println("No credit has been granted yet.");  
    }  
}  
}
```

iv. Code of BankGUI

```

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

public class BankGUI implements ActionListener
{
    //declare all the components here
    private JFrame frame;
    private ArrayList<BankCard> cards;

    private JLabel TitleText, DebitText, CreditText;

    //.....declare debit card components.....
    private JTextField D_CardIdText, D_ClientNameText, D_IssuerBankText,
    D_BankAccountText, D_BalanceAmountText;
    private JTextField D_PinNumberText, D_CardIDText, D_WithdrawalAmountText,
    D_PinNumberrText;

    private JLabel D_CardIdLabel, D_ClientNameLabel, D_IssuerBankLabel,
    D_BankAccountLabel, D_BalanceAmountLabel;
    private JLabel D_PinNumberLabel, D_CardIDLabel, D_WithdrawalAmountLabel,
    D_WithdrawalDateLabel, D_PinNumberrLabel;

    private JButton D_DisplayButton, D_AddButton, D_DisplayyButton,
    D_WithdrawButton;
    private JComboBox day, month, year;

```

```
//..... declare credit card components.....

private JTextField C_CardIdText, C_ClientNameText, C_IssuerBankText,
C_BankAccountText, C_BalanceAmountText;
private JTextField C_CVCNumberText, C_InterestRateText, C_CardIDText,
C_CreditLimitText, C_GracePeriodText, C_CarDIDText;

private JLabel C_CardIdLabel, C_ClientNameLabel, C_IssuerBankLabel,
C_BankAccountLabel, C_ExpirationDateLabel, C_BalanceAmountLabel;
private JLabel C_CVCNumberLabel, C_InterestRateLabel, C_CardIDLabel,
C_CreditLimitLabel, C_GracePeriodLabel, C_CarDIDLabel;

private JButton C_DisplayButton, C_AddButton, C_DisplayyButton, C_SetButton,
C_DisplayyyButton, C_CancelButton;
private JComboBox dayy, monthh, yearr;

private JButton ClearButton;

public BankGUI(){
cards = new ArrayList();

//create the code to write the GUI
//create frame
frame = new JFrame("Nabil Bank");
frame.setSize(1280,720);
frame.setLayout(null);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
frame.setResizable(true);

//create components
```

```
TitleText = new JLabel("Banking System");

//Debit Card components
//.....add to debit card.....

DebitText = new JLabel("Debit Card");

D_CardIdText = new JTextField();
D_CardIdLabel = new JLabel("Card Id:");

D_ClientNameText = new JTextField();
D_ClientNameLabel = new JLabel("Client Name:");

D_IssuerBankText = new JTextField();
D_IssuerBankLabel = new JLabel("Issuer Bank:");

D_BankAccountText = new JTextField();
D_BankAccountLabel = new JLabel("Bank Account:");

D_BalanceAmountText = new JTextField();
D_BalanceAmountLabel = new JLabel("Balance Amount:");

D_PinNumberText = new JTextField();
D_PinNumberLabel = new JLabel("Pin Number:");

D_DisplayButton = new JButton("Display");
D_DisplayButton.setBackground(new Color(212,227,252));
D_DisplayButton.setOpaque(true);
D_DisplayButton.setBorderPainted(true);
D_DisplayButton.addActionListener(this);

D_AddButton = new JButton("Add to Debit Card");
```

```

D_AddButton.setBackground(new Color(254,250,192));
D_AddButton.setOpaque(true);
D_AddButton.setBorderPainted(true);
D_AddButton.addActionListener(this);

//.....withdraw.....

D_CardIDText = new JTextField();
D_CardIDLabel = new JLabel("Card Id:");

D-WithdrawalAmountText = new JTextField();
D-WithdrawalAmountLabel = new JLabel("Withdrawal Amount:");

D-WithdrawalDateLabel = new JLabel("Withdrawal Date:");

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

String [] months = {"January","February","March","April",
"May","June","July","August","September","October",
"November","December"};
month = new JComboBox(months);

String [] years =
{"2011","2012","2013","2014","2015","2016","2017","2018","2019","2020",
"2021","2022","2023","2024","2025","2026","2027","2028","2029","2030"};
year = new JComboBox(years);

D_PinNumberrText = new JTextField();
D_PinNumberrLabel = new JLabel("Pin Number:");

```

```

D_DisplayyButton = new JButton("Display");
D_DisplayyButton.setBackground(new Color(212,227,252));
D_DisplayyButton.setOpaque(true);
D_DisplayyButton.setBorderPainted(true);
D_DisplayyButton.addActionListener(this);

D_WithdrawButton = new JButton("Withdraw");
D_WithdrawButton.setBackground(new Color(254,250,192));
D_WithdrawButton.setOpaque(true);
D_WithdrawButton.setBorderPainted(true);
D_WithdrawButton.addActionListener(this);

//Credit Card components
//.....add to credit card.....
CreditText = new JLabel("Credit Card");

C_CardIdText = new JTextField();
C_CardIdLabel = new JLabel("Card Id:");

C_ClientNameText = new JTextField();
C_ClientNameLabel = new JLabel("Client Name:");

C_IssuerBankText = new JTextField();
C_IssuerBankLabel = new JLabel("Issuer Bank:");

C_BankAccountText = new JTextField();
C_BankAccountLabel = new JLabel("Bank Account:");

C_ExpirationDateLabel = new JLabel("Expiration Date:");

```

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

```
String [] monthss = {"January","February","March","April",
"May","June","July","August","September","October",
"November","December"};
monthh = new JComboBox(monthss);
```

```
String [] yearss =
{"2011","2012","2013","2014","2015","2016","2017","2018","2019","2020",
"2021","2022","2023","2024","2025","2026","2027","2028","2029","2030"};
yarr = new JComboBox(yearss);
```

```
C_BalanceAmountText = new JTextField();
C_BalanceAmountLabel = new JLabel("Balance Amount:");
```

```
C_CVCNumberText = new JTextField();
C_CVCNumberLabel = new JLabel("CVC Number:");
```

```
C_InterestRateText = new JTextField();
C_InterestRateLabel = new JLabel("Interest Rate:");
```

```
C_DisplayButton = new JButton("Display");
C_DisplayButton.setBackground(new Color(212,227,252));
C_DisplayButton.setOpaque(true);
C_DisplayButton.setBorderPainted(true);
C_DisplayButton.addActionListener(this);
```

```
C_AddButton = new JButton("Add to Credit Card");
```



```

C_AddButton.setBackground(new Color(254,250,192));
C_AddButton.setOpaque(true);
C_AddButton.setBorderPainted(true);
C_AddButton.addActionListener(this);

//.....Set the credit limit.....
C_CardIDText = new JTextField();
C_CardIDLabel = new JLabel("Card Id:");

C_CreditLimitText = new JTextField();
C_CreditLimitLabel = new JLabel("Credit Limit:");

C_GracePeriodText = new JTextField();
C_GracePeriodLabel = new JLabel("Grace Period:");

C_DisplayyButton = new JButton("Display");
C_DisplayyButton.setBackground(new Color(212,227,252));
C_DisplayyButton.setOpaque(true);
C_DisplayyButton.setBorderPainted(true);
C_DisplayyButton.addActionListener(this);

C_SetButton = new JButton("Set Credit Limit");
C_SetButton.setBackground(new Color(254,250,192));
C_SetButton.setOpaque(true);
C_SetButton.setBorderPainted(true);
C_SetButton.addActionListener(this);

//.....Cancel Credit Card.....
C_CarDIDText = new JTextField();

```

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

```
C_DisplayyyButton = new JButton("Display");
```

```
C_DisplayyyButton.setBackground(new Color(212,227,252));
```

```
C_DisplayyyButton.setOpaque(true);
```

```
C_DisplayyyButton.setBorderPainted(true);
```

```
C_DisplayyyButton.addActionListener(this);
```

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

```
C_CancelButton.setBackground(new Color(254,250,192));
```

```
C_CancelButton.setOpaque(true);
```

```
C_CancelButton.setBorderPainted(true);
```

```
C_CancelButton.addActionListener(this);
```

```
ClearButton = new JButton("Clear");
```

```
ClearButton.setBackground(new Color(255,138,132));
```

```
ClearButton.setOpaque(true);
```

```
ClearButton.setBorderPainted(true);
```

```
ClearButton.addActionListener(this);
```

```
//set position
```

```
TitleText.setBounds(589, 10, 109, 20);
```

```
//.....Debit Card positions.....
```

```
DebitText.setBounds(298, 55, 71, 20);
```

```
D_CardIdLabel.setBounds(18, 104, 50, 18);
```

```
D_CardIdText.setBounds(150, 100, 142, 26);
```

```
D_ClientNameLabel.setBounds(18, 139, 81, 18);
```

```
D_ClientNameText.setBounds(150, 135, 142, 26);
```

```
D_IssuerBankLabel.setBounds(18, 174, 78, 18);  
D_IssuerBankText.setBounds(150, 170, 142, 26);
```

```
D_BankAccountLabel.setBounds(18, 209, 90, 18);  
D_BankAccountText.setBounds(150, 205, 142, 26);
```

```
D_BalanceAmountLabel.setBounds(336, 104, 106, 18);  
D_BalanceAmountText.setBounds(468, 100, 142, 26);
```

```
D_PinNumberLabel.setBounds(336, 139, 78, 18);  
D_PinNumberText.setBounds(468, 135, 142, 26);
```

```
D_DisplayButton.setBounds(319, 290, 142, 26);  
D_AddButton.setBounds(468, 290, 142, 26);
```

```
D_CardIDLabel.setBounds(18, 351, 50, 18);  
D_CardIDText.setBounds(150, 347, 142, 26);
```

```
D-WithdrawalAmountLabel.setBounds(18, 386, 125, 18);  
D-WithdrawalAmountText.setBounds(150, 382, 142, 26);
```

```
D-WithdrawalDateLabel.setBounds(18, 423, 107, 18);  
day.setBounds(150, 417, 40, 32);  
month.setBounds(200, 417, 90, 32);  
year.setBounds(300, 417, 60, 32);
```

```
D_PinNumberrLabel.setBounds(18, 460, 78, 18);  
D_PinNumberrText.setBounds(150, 457, 142, 26);
```

```
D_DisplayyButton.setBounds(319, 497, 142, 26);
```

```
D_WithdrawButton.setBounds(468, 497, 142, 26);

//.....Credit Card positions.....
CreditText.setBounds(936, 55, 71, 20);

C_CardIdLabel.setBounds(658, 104, 50, 18);
C_CardIdText.setBounds(790, 100, 142, 26);

C_ClientNameLabel.setBounds(658, 139, 81, 18);
C_ClientNameText.setBounds(790, 135, 142, 26);

C_IssuerBankLabel.setBounds(658, 174, 78, 18);
C_IssuerBankText.setBounds(790, 170, 142, 26);

C_BankAccountLabel.setBounds(658, 209, 87, 18);
C_BankAccountText.setBounds(790, 205, 142, 26);

C_ExpirationDateLabel.setBounds(658, 247, 100, 18);
dayy.setBounds(789, 240, 40, 32);
monthh.setBounds(839, 240, 90, 32);
yearr.setBounds(939, 240, 60, 32);

C_BalanceAmountLabel.setBounds(976, 104, 106, 18);
C_BalanceAmountText.setBounds(1108, 100, 142, 26);

C_CVCNumberLabel.setBounds(976, 139, 87, 18);
C_CVCNumberText.setBounds(1108, 135, 142, 26);

C_InterestRateLabel.setBounds(976, 174, 84, 18);
C_InterestRateText.setBounds(1108, 170, 142, 26);
```

```
C_DisplayButton.setBounds(959, 290, 142, 26);  
C_AddButton.setBounds(1108, 290, 142, 26);
```

```
C_CardIDLabel.setBounds(658, 351, 50, 18);  
C_CardIDText.setBounds(790, 347, 142, 26);
```

```
C_CreditLimitLabel.setBounds(658, 386, 75, 18);  
C_CreditLimitText.setBounds(790, 382, 142, 26);
```

```
C_GracePeriodLabel.setBounds(658, 421, 86, 18);  
C_GracePeriodText.setBounds(790, 417, 142, 26);
```

```
C_DisplayyButton.setBounds(959, 497, 142, 26);  
C_SetButton.setBounds(1108, 497, 142, 26);
```

```
C_CarDIDLabel.setBounds(658, 558, 50, 18);  
C_CarDIDText.setBounds(790, 554, 142, 26);
```

```
C_DisplayyyButton.setBounds(959, 592, 142, 26);  
C_CancelButton.setBounds(1108, 592, 143, 26);
```

```
ClearButton.setBounds(586, 615, 142, 26);
```

```
//add componenet  
frame.add(TitleText);
```

```
//.....add componenets of Debit Card.....  
frame.add(DebitText);  
  
frame.add(D_CardIdText);  
frame.add(D_CardIdLabel);  
  
frame.add(D_ClientNameLabel);  
frame.add(D_ClientNameText);  
  
frame.add(D_IssuerBankLabel);  
frame.add(D_IssuerBankText);  
  
frame.add(D_BankAccountLabel);  
frame.add(D_BankAccountText);  
  
frame.add(D_BalanceAmountLabel);  
frame.add(D_BalanceAmountText);  
  
frame.add(D_PinNumberText);  
frame.add(D_PinNumberLabel);  
  
frame.add(D_DisplayButton);  
frame.add(D_AddButton);  
  
frame.add(D_CardIDText);  
frame.add(D_CardIDLabel);  
  
frame.add(D-WithdrawalAmountLabel);  
frame.add(D-WithdrawalAmountText);
```

```
frame.add(D_WithdrawalDateLabel);
frame.add(day);
frame.add(month);
frame.add(year);

frame.add(D_PinNumbeerrText);
frame.add(D_PinNumbeerrLabel);

frame.add(D_DisplayyButton);
frame.add(D_WithdrawButton);

//.....add componenets of Credit Card.....
frame.add(CreditText);

frame.add(C_CardIdText);
frame.add(C_CardIdLabel);

frame.add(C_ClientNameLabel);
frame.add(C_ClientNameText);

frame.add(C_IssuerBankLabel);
frame.add(C_IssuerBankText);

frame.add(C_BankAccountLabel);
frame.add(C_BankAccountText);

frame.add(C_ExpirationDateLabel);
frame.add(dayy);
frame.add(monthh);
frame.add(yearr);
```

```
frame.add(C_BalanceAmountLabel);  
frame.add(C_BalanceAmountText);
```

```
frame.add(C_CVCNumberLabel);  
frame.add(C_CVCNumberText);
```

```
frame.add(C_InterestRateLabel);  
frame.add(C_InterestRateText);
```

```
frame.add(C_DisplayButton);  
frame.add(C_AddButton);
```

```
frame.add(C_CardIDText);  
frame.add(C_CardIDLabel);
```

```
frame.add(C_CreditLimitLabel);  
frame.add(C_CreditLimitText);
```

```
frame.add(C_GracePeriodLabel);  
frame.add(C_GracePeriodText);
```

```
frame.add(C_DisplayyButton);  
frame.add(C_SetButton);
```

```
frame.add(C_CarDIDText);  
frame.add(C_CarDIDLabel);
```

```
frame.add(C_DisplayyyyButton);  
frame.add(C_CancelButton);
```



```

frame.add(ClearButton);
}

//implement the method of the ActionListener
public void actionPerformed(ActionEvent e){
//logic of the button functionality
//.....ADD Debit.....
if (e.getSource() == D_AddButton){
if(D_CardIdText.getText().isEmpty() || D_ClientNameText.getText().isEmpty() ||
D_IssuerBankText.getText().isEmpty() ||
D_BankAccountText.getText().isEmpty() ||
D_BalanceAmountText.getText().isEmpty() ||
D_PinNumberText.getText().isEmpty()){
JOptionPane.showMessageDialog(frame, "The fields are empty !!!");
}
else{
try{
String clientName = D_ClientNameText.getText();
String issuerBank = D_IssuerBankText.getText();
String bankAccount = D_BankAccountText.getText();

int cardId = Integer.parseInt(D_CardIdText.getText());
int balanceAmount = Integer.parseInt(D_BalanceAmountText.getText());
int pinNumber = Integer.parseInt(D_PinNumberText.getText());

boolean Existence_of_Card = false;
for(BankCard DebitCard : cards){
if(DebitCard instanceof DebitCard){
DebitCard DebitCard_Object = (DebitCard) DebitCard;

```

```

if(DebitCard_Object.getCardId() == (cardId))    //Comparing card's id from
ArrayList
{
Existence_of_Card = true;
}
}
}
if(Existence_of_Card == false){
DebitCard DebitCard_Object= new DebitCard(balanceAmount, cardId,
bankAccount, issuerBank, clientName, pinNumber); //should be in the order of
DebitCard constructor
cards.add(DebitCard_Object);
JOptionPane.showMessageDialog(frame, "Your Debit Card has been added
successfully!!!");
}else{
JOptionPane.showMessageDialog(frame, "Debit Card already exists!!!");
}
}
catch(NumberFormatException nfe){
JOptionPane.showMessageDialog(frame,"Please provide required information
only!!!");
}
}
}

//.....Display for ADD Debit.....
if(e.getSource() == D_DisplayButton){
if(D_CardIdText.getText().isEmpty() || D_ClientNameText.getText().isEmpty() ||
D_IssuerBankText.getText().isEmpty() ||

```

```

D_BankAccountText.getText().isEmpty() ||
D_BalanceAmountText.getText().isEmpty() ||
D_PinNumberText.getText().isEmpty()){
JOptionPane.showMessageDialog(frame, "The fields are empty !!");
}
else{
try{
String clientName = D_ClientNameText.getText();
String issuerBank = D_IssuerBankText.getText();
String bankAccount = D_BankAccountText.getText();

int cardId = Integer.parseInt(D_CardIdText.getText());
int balanceAmount = Integer.parseInt(D_BalanceAmountText.getText());
int pinNumber = Integer.parseInt(D_PinNumberText.getText());

for(BankCard Display_DebitCard : cards){
if(Display_DebitCard instanceof DebitCard){
DebitCard DebitCard_Object = (DebitCard) Display_DebitCard;
DebitCard_Object.display();
JOptionPane.showMessageDialog(frame, "Debit Card information are displayed.
Please check the display tab!!!");
}
}
}
catch(NumberFormatException nfe){
JOptionPane.showMessageDialog(frame, "Please provide required information
only!!!");
}
}
}

```

```
//.....Withdraw Button.....
if (e.getSource() == D_WithdrawButton){
    if(D_CardIDText.getText().isEmpty() ||
    D_WithdrawalAmountText.getText().isEmpty() ||
    D_WithdrawalDateLabel.getText().isEmpty() ||
    D_PinNumberrText.getText().isEmpty()){
        JOptionPane.showMessageDialog(frame, "The fields are empty !!");
    }
    else{
        try{
            String dateOfWithdrawal = D_WithdrawalDateLabel.getText();

            int cardID = Integer.parseInt(D_CardIDText.getText());
            int pinNumber = Integer.parseInt(D_PinNumberrText.getText());
            int withdrawalAmount = Integer.parseInt(D_WithdrawalAmountText.getText());

            for(BankCard Withdraw_DebitCard : cards){
                if(Withdraw_DebitCard instanceof DebitCard){
                    DebitCard DebitCard_Object= (DebitCard) Withdraw_DebitCard;
                    if(DebitCard_Object.getCardId() == (cardID)){
                        DebitCard_Object.withdraw(withdrawalAmount, dateOfWithdrawal, pinNumber);
                        JOptionPane.showMessageDialog(frame,"Your Debit Card has been withdrawn
                        successfully!!!");
                    }else{
                        JOptionPane.showMessageDialog(frame,"Debit Card doesn't exist");
                    }
                }
            }
        }
        catch(NumberFormatException nfe){
```

```

JOptionPane.showMessageDialog(frame, "Please provide required information
only!!!");
}
}
}

```

```

//.....Display for Withdraw Debit.....
if(e.getSource() == D_DisplayButton){
if(D_CardIdText.getText().isEmpty() || D_ClientNameText.getText().isEmpty() ||
D_IssuerBankText.getText().isEmpty() ||
D_BankAccountText.getText().isEmpty() ||
D_BalanceAmountText.getText().isEmpty() ||
D_PinNumberText.getText().isEmpty() ||
D_CardIDText.getText().isEmpty() ||
D_WithdrawalAmountText.getText().isEmpty() ||
D_WithdrawalDateLabel.getText().isEmpty() ||
D_PinNumberText.getText().isEmpty()){
JOptionPane.showMessageDialog(frame, "The fields are empty !!");
}
else{
try{
String clientName = D_ClientNameText.getText();
String issuerBank = D_IssuerBankText.getText();
String bankAccount = D_BankAccountText.getText();
String dateOfWithdrawal = D_WithdrawalDateLabel.getText();

int cardId = Integer.parseInt(D_CardIdText.getText());
int balanceAmount = Integer.parseInt(D_BalanceAmountText.getText());
int pinNumber = Integer.parseInt(D_PinNumberText.getText());
int cardID = Integer.parseInt(D_CardIDText.getText());

```

```

int pinNumberR = Integer.parseInt(D_PinNumberrText.getText());
int withdrawalAmount = Integer.parseInt(D-WithdrawalAmountText.getText());

for(BankCard Displayy_DebitCard : cards){
    if(Displayy_DebitCard instanceof DebitCard){
        DebitCard DebitCard_Object= (DebitCard) Displayy_DebitCard;
        DebitCard_Object.display();
        JOptionPane.showMessageDialog(frame, "All Debit Card informations are
        displayed. Please check the display tab!!!");
    }
}

catch(NumberFormatException nfe){
    JOptionPane.showMessageDialog(frame,"Please provide required informations
    only!!!");
}

}

}

//.....ADD Credit.....
if (e.getSource() == C_AddButton){
    if(C_CardIdText.getText().isEmpty() || C_ClientNameText.getText().isEmpty() ||
    C_IssuerBankText.getText().isEmpty() ||
    C_BankAccountText.getText().isEmpty() ||
    C_BalanceAmountText.getText().isEmpty() ||
    C_CVCNumberText.getText().isEmpty() ||
    C_InterestRateText.getText().isEmpty() ||
    C_ExpirationDateLabel.getText().isEmpty()){
        JOptionPane.showMessageDialog(frame, "The fields are empty !!");
    }
}

```

```

else{
try{
String clientName = C_ClientNameText.getText();
String issuerBank = C_IssuerBankText.getText();
String bankAccount = C_BankAccountText.getText();
String expirationDate = C_ExpirationDateLabel.getText();

int cardId = Integer.parseInt(C_CardIdText.getText());
int balanceAmount = Integer.parseInt(C_BalanceAmountText.getText());
int cvcNumber = Integer.parseInt(C_CVCNumberText.getText());

double interestRate = Double.parseDouble(C_InterestRateText.getText());

boolean Existence_of_Card = false;
for(BankCard CreditCard : cards){
if(CreditCard instanceof CreditCard){
CreditCard CreditCard_Object = (CreditCard) CreditCard;
if(CreditCard_Object.getCardId() == cardId)    //Comparing card's id from
ArrayList
{
Existence_of_Card = true;
}
}
}
}
//Make a new debit card if the card don't exist
if(Existence_of_Card == false){
CreditCard CreditCard_Object= new CreditCard(balanceAmount, cardId,
bankAccount, issuerBank, clientName, cvcNumber, interestRate, expirationDate);
//should be in the order of DebitCard constructor
cards.add(CreditCard_Object);
}
}
}

```

```

JOptionPane.showMessageDialog(frame, "Your Credit Card has been added
successfully!!!");
}else{
//Display the message about existence of debit card if the card exist
JOptionPane.showMessageDialog(frame, "Credit Card already exists.");
}
}
catch(NumberFormatException nfe){
JOptionPane.showMessageDialog(frame, "Please provide required information
only!!!");
}
}
}

//.....Display for ADD Credit.....
if(e.getSource() == C_DisplayButton){
if(C_CardIdText.getText().isEmpty() || C_ClientNameText.getText().isEmpty() ||
C_IssuerBankText.getText().isEmpty() ||
C_BankAccountText.getText().isEmpty() ||
C_BalanceAmountText.getText().isEmpty() ||
C_CVCNumberText.getText().isEmpty() ||
C_InterestRateText.getText().isEmpty() ||
C_ExpirationDateLabel.getText().isEmpty()){
JOptionPane.showMessageDialog(frame, "The fields are empty !!");
}
else{
try{
String clientName = C_ClientNameText.getText();
String issuerBank = C_IssuerBankText.getText();
String bankAccount = C_BankAccountText.getText();

```



```

String expirationDate = C_ExpirationDateLabel.getText();

int cardId = Integer.parseInt(C_CardIdText.getText());
int balanceAmount = Integer.parseInt(C_BalanceAmountText.getText());
int cvcNumber = Integer.parseInt(C_CVCNumberText.getText());

double interestRate = Double.parseDouble(C_InterestRateText.getText());

for(BankCard Display_CreditCard : cards){
    if(Display_CreditCard instanceof CreditCard){
        CreditCard CreditCard_Object= (CreditCard) Display_CreditCard;
        CreditCard_Object.display();
        JOptionPane.showMessageDialog(frame, "Credit Card information are displayed.
        Please check the display tab!!!");
    }
}

catch(NumberFormatException nfe){
    JOptionPane.showMessageDialog(frame, "Please provide required informations
    only!!!");
}

}

}

//.....Set the Credit Limit.....
if (e.getSource() == C_SetButton){
    if(C_CardIdText.getText().isEmpty() || C_CreditLimitText.getText().isEmpty() ||
    C_GracePeriodText.getText().isEmpty()){
        JOptionPane.showMessageDialog(frame, "The fields are empty !!");
    }
}

```

```

else{
try{
int cardID = Integer.parseInt(C_CardIDText.getText());
int gracePeriod = Integer.parseInt(C_GracePeriodText.getText());

double creditLimit = Double.parseDouble(C_CreditLimitText.getText());

for(BankCard Set_CreditCard : cards){
if(Set_CreditCard instanceof CreditCard){
CreditCard CreditCard_Object = (CreditCard) Set_CreditCard;
if(CreditCard_Object.getCardId() == (cardID)){
if(creditLimit <= (CreditCard_Object.getBalanceAmount() * 2.5)){
CreditCard_Object.setCreditLimit(creditLimit, gracePeriod);
JOptionPane.showMessageDialog(frame, "Your Credit Limit has been set
successfully!!!");

}
else{
JOptionPane.showMessageDialog(frame, "Credit Card has been exceeded.
Please try again!!!");
}
}
else{
JOptionPane.showMessageDialog(frame, "Credit Card doesn't exist.");
}
}
}
}

catch(NumberFormatException nfe){
JOptionPane.showMessageDialog(frame, "Invalid values !!! Please enter correct
values.");
}
}

```

```

}
}
}

```

```

//.....Display for Set Credit Limit.....
if(e.getSource() == C_DisplayyButton){
if(C_CardIdText.getText().isEmpty() || C_ClientNameText.getText().isEmpty() ||
C_IssuerBankText.getText().isEmpty() ||
C_BankAccountText.getText().isEmpty() ||
C_BalanceAmountText.getText().isEmpty() ||
C_CVCNumberText.getText().isEmpty() ||
C_InterestRateText.getText().isEmpty() ||
C_ExpirationDateLabel.getText().isEmpty() || C_CardIDText.getText().isEmpty() ||
C_CreditLimitText.getText().isEmpty() || C_GracePeriodText.getText().isEmpty()){
JOptionPane.showMessageDialog(frame, "The fields are empty !!");
}
else{
try{
String clientName = C_ClientNameText.getText();
String issuerBank = C_IssuerBankText.getText();
String bankAccount = C_BankAccountText.getText();
String expirationDate = C_ExpirationDateLabel.getText();

int cardId = Integer.parseInt(C_CardIdText.getText());
int balanceAmount = Integer.parseInt(C_BalanceAmountText.getText());
int cardID = Integer.parseInt(C_CardIDText.getText());
int cvcNumber = Integer.parseInt(C_CVCNumberText.getText());
int gracePeriod = Integer.parseInt(C_GracePeriodText.getText());

double creditLimit = Double.parseDouble(C_CreditLimitText.getText());

```

```

double interestRate = Double.parseDouble(C_InterestRateText.getText());

for(BankCard Displayy_CreditCard : cards){
    if(Displayy_CreditCard instanceof CreditCard){
        CreditCard CreditCard_Object= (CreditCard) Displayy_CreditCard;
        CreditCard_Object.display();
        JOptionPane.showMessageDialog(frame, "Credit Card information for setting the
        Credit Limit are displayed. Please check the display tab!!!");
    }
}

catch(NumberFormatException nfe){
    JOptionPane.showMessageDialog(frame,"Please provide required informations
    only!!!");
}

//.....Cancel Credit Card Button.....
if (e.getSource() == C_CancelButton){
    if(C_CarDIDText.getText().isEmpty()){
        JOptionPane.showMessageDialog(frame, "The fields are empty !!");
    }
    else{
        try{
            int carDID = Integer.parseInt(C_CarDIDText.getText());

            for(BankCard Cancel_CreditCard : cards){
                if(Cancel_CreditCard instanceof CreditCard){
                    CreditCard CreditCard_Object = (CreditCard) Cancel_CreditCard;

```

```

if(CreditCard_Object.getCardId() == (carDID)){
CreditCard_Object.cancelCreditCard();
JOptionPane.showMessageDialog(frame, "Credit Card has been cancelled
successfully!!!");
}else{
JOptionPane.showMessageDialog(frame, "Credit Card hasn't been cancelled");
}
}
}
}
}
catch(NumberFormatException nfe){
JOptionPane.showMessageDialog(frame, "Please provide required information
only!!!");
}
}
}

//.....Display for Cancel Credit
Card.....
if(e.getSource() == C_DisplayyyyButton){
if(C_CardIdText.getText().isEmpty() || C_ClientNameText.getText().isEmpty() ||
C_IssuerBankText.getText().isEmpty() ||
C_BankAccountText.getText().isEmpty() ||
C_BalanceAmountText.getText().isEmpty() ||
C_CVCNumberText.getText().isEmpty() ||
C_InterestRateText.getText().isEmpty() ||
C_ExpirationDateLabel.getText().isEmpty() || C_CardIDText.getText().isEmpty() ||
C_CreditLimitText.getText().isEmpty() || C_GracePeriodText.getText().isEmpty() ||
C_CarDIDText.getText().isEmpty()){
JOptionPane.showMessageDialog(frame, "The fields are empty !!!");
}
}
}

```

```

}
else{
try{
String clientName = C_ClientNameText.getText();
String issuerBank = C_IssuerBankText.getText();
String bankAccount = C_BankAccountText.getText();
String expirationDate = C_ExpirationDateLabel.getText();

int cardId = Integer.parseInt(C_CardIdText.getText());
int balanceAmount = Integer.parseInt(C_BalanceAmountText.getText());
int cardID = Integer.parseInt(C_CardIDText.getText());
int cvcNumber = Integer.parseInt(C_CVCNumberText.getText());
int gracePeriod = Integer.parseInt(C_GracePeriodText.getText());
int carDID = Integer.parseInt(C_CarDIDText.getText());

double creditLimit = Double.parseDouble(C_CreditLimitText.getText());
double interestRate = Double.parseDouble(C_InterestRateText.getText());

for(BankCard Displayyy_CreditCard : cards){
if(Displayyy_CreditCard instanceof CreditCard){
CreditCard CreditCard_Object= (CreditCard) Displayyy_CreditCard;
CreditCard_Object.display();
JOptionPane.showMessageDialog(frame, "All Credit Card informations are
displayed. Please check the display tab!!!");
}
}
}
catch(NumberFormatException nfe){
JOptionPane.showMessageDialog(frame,"Please provide required informations
only!!!");
}
}

```

```

}
}

//.....Clear Button.....
if (e.getSource() == ClearButton){
    D_CardIdText.setText("");
    D_ClientNameText.setText("");
    D_IssuerBankText.setText("");
    D_BankAccountText.setText("");
    D_BalanceAmountText.setText("");
    D_PinNumberText.setText("");

    D_CardIDText.setText("");
    D_WithdrawalAmountText.setText("");
    D_PinNumberrText.setText("");

    C_CardIdText.setText("");
    C_ClientNameText.setText("");
    C_IssuerBankText.setText("");
    C_BankAccountText.setText("");
    C_BalanceAmountText.setText("");
    C_CVCNumberText.setText("");
    C_InterestRateText.setText("");

    C_CardIDText.setText("");
    C_GracePeriodText.setText("");
    C_CreditLimitText.setText("");

    C_CarDIDText.setText("");
}

```

```
}
```

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