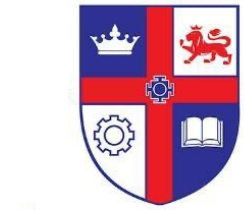
LONDON METROPOLITAN UNIVERSITY



islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Bidhan Shrestha

London Met ID: 22067584

College ID: NP01CP4A220077

Group: C4

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 mark of zero will be awarded.

Table of Contents

Table of Figures

Table of Tables

1.Introduction:

1.1 Brief Introduction:

The BankCard, DebitCard, and CreditCard classes are discussed. These classes will represent the various credit card types that are used for transactions. The Bankcard class will stand in for a basic card with several functions, including cash withdrawals and transactions. The DebitCard subclass of the Bankcard class will be able to perform operations like cash withdrawals. The CreditCard class, which is also a subclass of the Bankcard class, will apply to a card that allows the cardholder to borrow money from the bank and make payments later.

To store details about the card, such the card number, expiration date, and cardholder name, each class will include instance variables. Additionally, they will detail how to carry out other tasks including confirming card data, paying bills, and monitoring the card balance. The lectures will also cover standard techniques for verifying card data, determining available balances, and creating transaction reports.

2 Tools used:

1.1.1 Blue J:

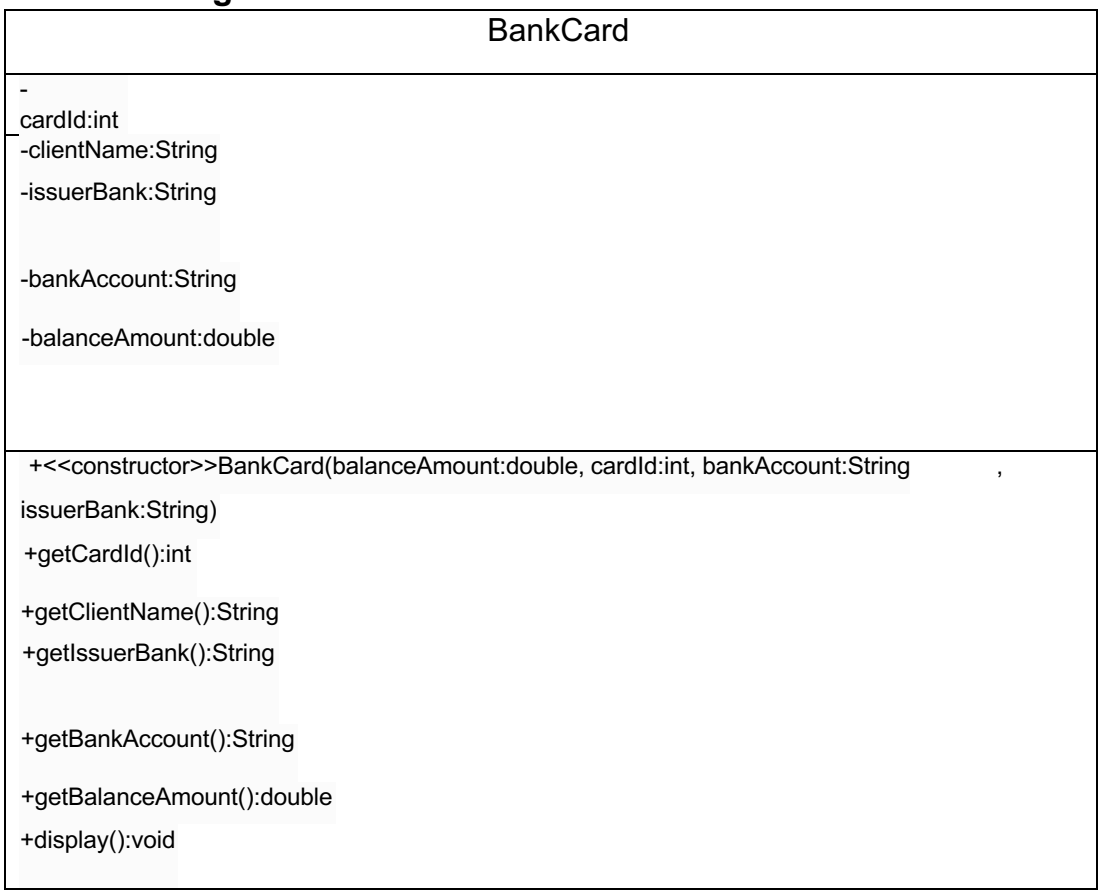
BlueJ is a development environment that makes it simple and quick to create Java apps. Its primary characteristics include the following Basic Compared to professional environments like NetBeans or Eclipse, BlueJ's UI is purposefully smaller and simpler. (Anonymous, n.d.)

1.1.2 Draw io:

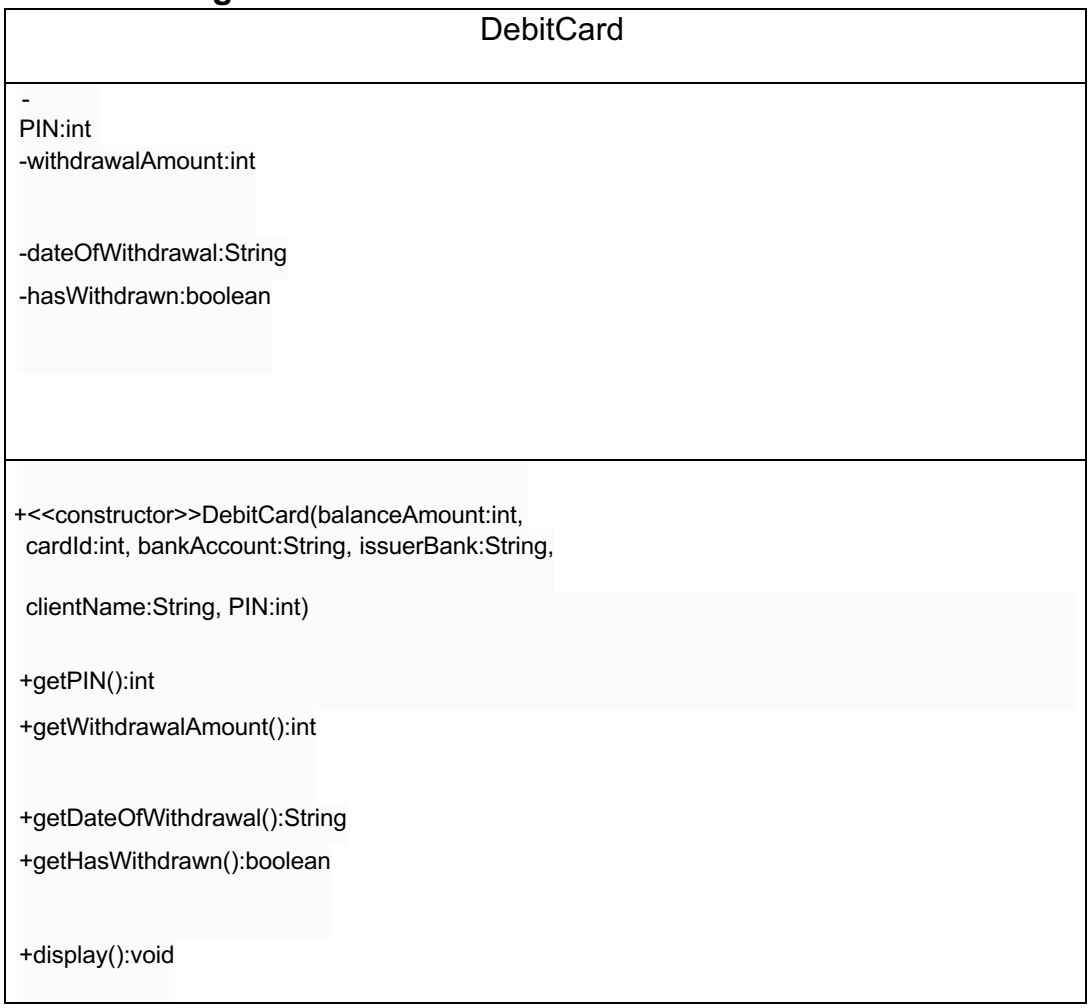
A cross-platform, free and open source graph sketching program called Draw io was created in HTML5 and JavaScript. Wireframes, UML diagrams, organizational charts, network diagrams, and flowcharts may all be made using its interface. (Anonymous, n.d.)

2. Class Diagram:

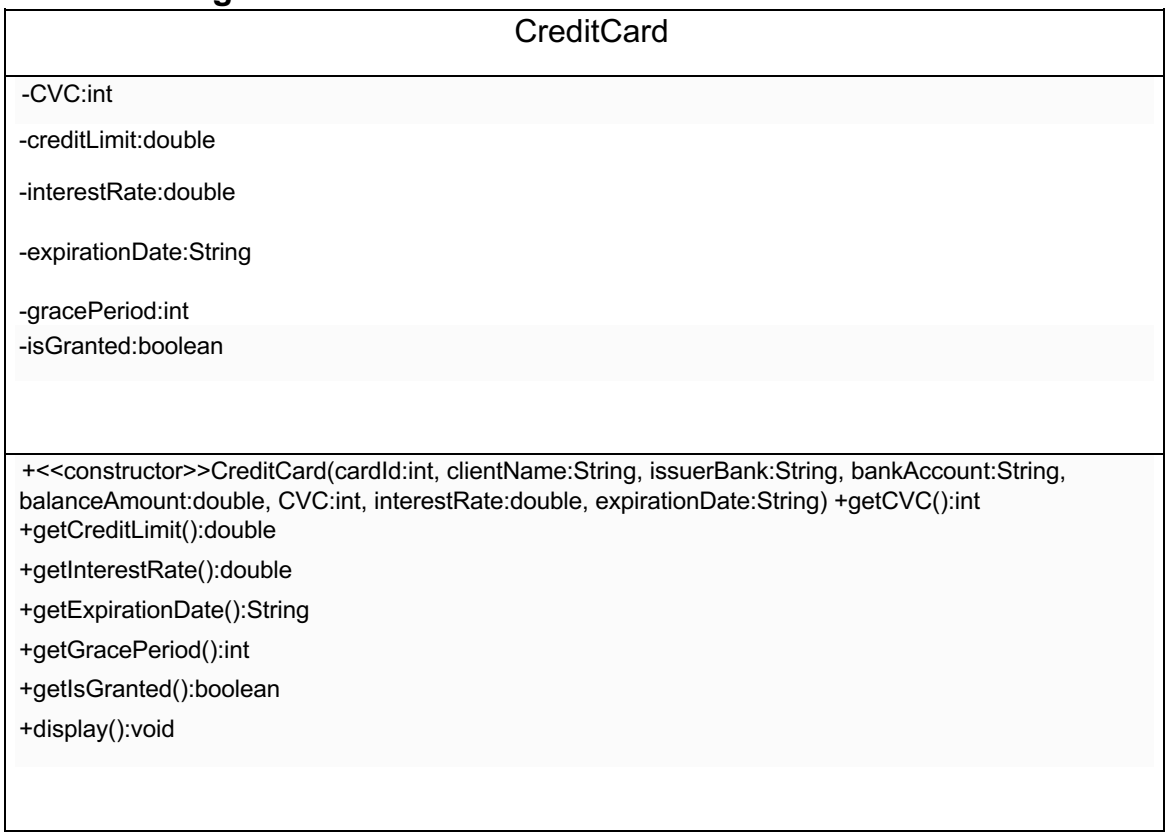
2.1 Class Diagram of Bank Card:



2.2 Class Diagram of Debit Card:



2.3 Class Diagram of Credit Card:



2.4 Class Diagram of BankGUI:

BankGUI

-frame: Jframe
- headingtext, Debitheading, Creditheading- Jlabel

- label, DebitCard, CreditCard, DCardID, DClientname, DBankAccount, DIssuerBank, DBalanceAmount,
DPinNumber, CCardID, CCVC, CBankAccount, CIssuerBank, CBalanceAmount, CClientName, CInterestRate, CExpiraFonDate, DebitCardID, DebitWithdrawalAmount, DebitWithdrawalDate, DebitPinNumber, CreditCardID, CreditCreditLimit, CreditGracePeriod- JLabel

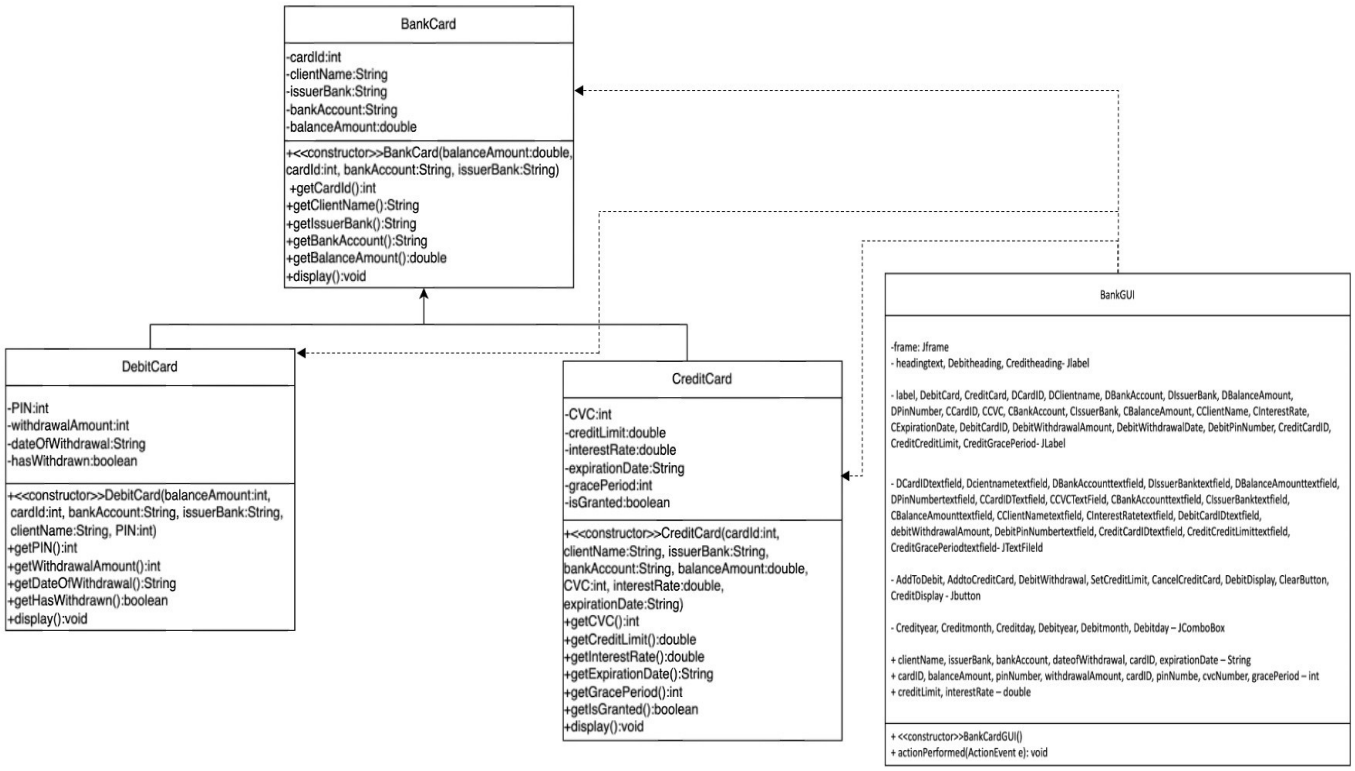
- DCardIDtexJield, DcientnametexJield, DBankAccounKexJield, DIssuerBanktexJield, DBalanceAmounKexJield, DPinNumbertexJield, CCardIDTexJield, CCVCTextField, CBankAccounKexJield, CIssuerBanktexJield, CBalanceAmounKexJield, CClientNametexJield, CInterestRatetexJield, DebitCardIDtexJield, debitWithdrawalAmount, DebitPinNumbertexJield, CreditCardIDtexJield, CreditCreditLimiKexJield, CreditGracePeriodtexJield- JTextFiIeld

- AddToDebit, AddtoCreditCard, DebitWithdrawal, SetCreditLimit, CancelCreditCard, DebitDisplay, ClearBuKon, CreditDisplay - JbuKon

- Crediтyear, Creditmonth, Creditday, Debityear, Debitmonth, Debitday – JComboBox

+ clientName, issuerBank, bankAccount, dateofWithdrawal, cardID, expiraFonDate – String
+ cardID, balanceAmount, pinNumber, withdrawalAmount, cardID, pinNumbe, cvcNumber, gracePeriod – int
+ creditLimit, interestRate – double

+ <<constructor>>BankCardGUI()
+ acFonPerformed(AcFonEvent e): void


BankCard
-cardId:int
-clientName:String
-issuerBank:String
-bankAccount:String
-balanceAmount:double
+<<constructor>>BankCard(balanceAmount:double, cardId:int, bankAccount:String, issuerBank:String)
+getCardId():int
+getClientName():String
+getIssuerBank():String
+getBankAccount():String
+getBalanceAmount():double
+display():void
DebitCard
-PIN:int
-withdrawalAmount:int
-dateOfWithdrawal:String
-hasWithdrawn:boolean
+<<constructor>>DebitCard(balanceAmount:int, cardId:int, bankAccount:String, issuerBank:String, clientName:String, PIN:int)
+getPIN():int
+getWithdrawalAmount():int
+getDateOfWithdrawal():String
+getHasWithdrawn():boolean
+display():void
CreditCard
-CVC:int
-creditLimit:double
-interestRate:double
-expirationDate:String
-gracePeriod:int
-isGranted:boolean
+<<constructor>>CreditCard(cardId:int, clientName:String, issuerBank:String, bankAccount:String, balanceAmount:double, CVC:int, interestRate:double, expirationDate:String)
+getCVC():int
+getCreditLimit():double
+getInterestRate():double
+getExpirationDate():String
+getGracePeriod():int
+getIsGranted():boolean
+display():void
BankGUI
-frame: Jframe
- headingtext, Debitheading, Creditheading- Jlabel
- label, DebitCard, CreditCard, DCardID, DClientname, DBankAccount, DIssuerBank, DBalanceAmount, DPinNumber, CCardID, CCVC, CBankAccount, CIssuerBank, CBalanceAmount, CClientName, CInterestRate, CExpirationDate, DebitCardID, DebitWithdrawalAmount, DebitWithdrawalDate, DebitPinNumber, CreditCardID, CreditCreditLimit, CreditGracePeriod- JLabel
- DCardIDtextfield, Dcientnametextfield, DBankAccounttextfield, DIssuerBanktextfield, DBalanceAmounttextfield, DPinNumbertextfield, CCardIDTextfield, CCVCTextField, CBankAccounttextfield, CIssuerBanktextfield, CBalanceAmounttextfield, CClientNametextfield, CInterestRatetextfield, DebitCardIDtextfield, debitWithdrawalAmount, DebitPinNumbertextfield, CreditCardIDtextfield, CreditCreditLimittextfield, CreditGracePeriodtextfield- JTextFileld
- AddToDebit, AddtoCreditCard, DebitWithdrawal, SetCreditLimit, CancelCreditCard, DebitDisplay, ClearButton, CreditDisplay - Jbutton
- Crediyear, Creditmonth, Creditday, Debityear, Debitmonth, Debitday – JComboBox
+ clientName, issuerBank, bankAccount, dateofWithdrawal, cardID, expirationDate – String
+ cardID, balanceAmount, pinNumber, withdrawalAmount, cardID, pinNumbe, cvcNumber, gracePeriod – int
+ creditLimit, interestRate – double
+ <<constructor>>BankCardGUI()
+ actionPerformed(ActionEvent e): void

3. Pseudocode

3.1 Pseudocode of BankGUI:

IMPORT javax.swing.*\

IMPORT java.awt.*;

IMPORT java.awt.event.ActionEvent;

IMPORT java.awt.event.ActionListener;

IMPORT java.util.ArrayList;

CREATE a class BAnkGUI which implements ActionListener

DO

DECLARE instance variable cardList as ArrayList <BankCard>

DECLARE instance variable label, DebitCard, CreditCard, DCardID, DClientname, DBankAccount, DIssuerBank, DBalanceAmount, DPinNumber, CCardID, CCVC, CBankAccount,

CIssuerBank, CBalanceAmount, CClientName, CInterestRate, CExpirationDate, DebitCardID, DebitWithdrawalAmount, DebitWithdrawalDate, DebitPinNumber, CreditCardID, CreditCreditLimit, CreditGracePeriod as JLabel using private access modifier.

DECLARE DCardIDtextfield, Dcientnametextfield, DBankAccounttextfield, DIssuerBanktextfield, DBalanceAmounttextfield, DPinNumbertextfield, CCardIDTextfield, CCVCTextField, CBankAccounttextfield, CIssuerBanktextfield, CBalanceAmounttextfield, CClientNametextfield, CInterestRatetextfield, DebitCardIDtextfield, debitWithdrawalAmount, DebitPinNumbertextfield, CreditCardIDtextfield,CreditCreditLimittextfield, CreditGracePeriodtextfield as JTextField using private access modifier.

DECLARE AddToDebit, AddtoCreditCard, DebitWithdrawal, SetCreditLimit, CancelCreditCard, DebitDisplay, ClearButton, CreditDisplay as JButton using private access modifier.

DECLARE Crediyear, Creditmonth, Creditday, Debityear, Debitmonth, Debitday as JComboBox using private access modifier.

CREATE method BankGUI with return type void and no parameters

DO

INITILIAZE cardList as ArrayList

INITILIAZE frame as JFrame

SET the background color of the frame

DECLARE the required number of JLabel for DebitCard

DECLARE the required number of JTextField for DebitCard

DECLARE the required number of JButton for DebitCard

DECLARE the required number of JComboBox for DebitCard

DECLARE the required number of JLabel for CreditCard

DECLARE the required number of JTextField for CreditCard

DECLARE the required number of JButton for CreditCard

DECLARE the required number of JComboBox for CreditCard

SET required bounds for JFrame

SET bounds for required number of JLabel for Debit Card

SET bounds for required number of JTextField for Debit Card

SET bounds for required number of JButton for Debit Card

SET bounds for required number of JComboBox for Debit Card

SET bounds for required number of JLabel for Credit Card

SET bounds for required number of JTextField for Credit Card

SET bounds for required number of JButton for Credit Card

SET bounds for required number of JComboBox for Credit Card

ADD required number of JLabel for Debit Card

ADD required number of JTextField for Debit Card

ADD required number of JButton for Debit Card

ADD required number of JComboBox for Debit Card

ADD required number of JLabel for Credit Card

ADD required number of JTextField for Credit Card

ADD required number of JButton for Credit Card

ADD required number of JComboBox for Credit Card

ADD action listener to the required buttons

SET resizable none

SET Background color

SET layout to null

SET frame visibility to true

END DO

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

DO

IF e.getSource() is AddToDebit

DO

IF DBalanceAmounttextfieldgetText() is empty or DCardIDtextfield.getText()is empty or DBankAccounttextfield.getText() is empty or DIssuerBanktextfield.getText() is empty or Dcientnametextfield.getText() is empty or DPinNumbertextfield.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 Dcientnametextfield.getText()

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

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

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

DECLARE balanceAmount as int with value of DBalanceAmounttextfield.getText() **DECLARE** pinNumber as int with value of DPinNumbertextfieldgetText()

CREATE new debit card with balance, card ID, bank account, issuer bank, client name, and PIN

ADD newCard to cardList

DISPLAY Debit Card added on message dialogue

END DO

CATCH

DO

DISPLAY PLEASE ENTER CORRECT VALUES on message dialogue

END DO

END DO

END DO

DO

IF e.getSource() is AddtoCreditCard

DO

IF CCardIDTextfield.getText() is empty or CCVCTextField.getText()is empty or CIssuerBanktextfield.getText() is empty or CBankAccounttextfield.getText() is empty or

CBalanceAmounttextfield.getText() is empty or CBalanceAmounttextfield.getText() is empty or CClientNametextfield.getText() is empty or CExpirationDate.getText() is empty or

CInterestRatetextfield.getText() is empty

DO

DISPLAY THE FIELDS ARE EMPTY on message dialogue

END DO

ELSE

DO

TRY

DO

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

DECLARE cvc as int with value of CCVCTextField.getText()

DECLARE issuerBankas string with value of CIssuerBanktextfield.getText()

DECLARE bankAccount string with value of CBankAccounttextfield.getText()

DECLARE balance as int with value of CBalanceAmounttextfield.getText()

DECLARE clientName string with value of CClientNametextfield.getText()

DECLARE expirationDate string with value of CExpirationDate.getText()

DECLARE interestRate double with value of CInterestRatetextfield.getText()

CREATE new credit card with card ID, client name, issuer bank, bank account, balance, CVC, interest rate, and expiration date, and assign it to newCard

ADD newCard to cardList

DISPLAY Credit Card added! on message dialogue

END DO

CATCH

DISPLAY PLEASE ENTER CORRECT VALUES on message dialogue

END DO

END DO

END DO

END DO

IF e.getSource() is DebitWithdrawal

DO

IF DCardIDtextfield.getText() is empty or debitWithdrawalAmount.getText() is empty or DebitWithdrawalDate.getText() is empty or DebitPinNumbertextfield.getText() is empty

DO

DISPLAY THE FIELDS ARE EMPTY on message dialogue

END DO

ELSE

DO

TRY

DO

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

DECLARE withdrawalAmount as int with value of debitWithdrawalAmount.getText()

DECLARE dateOfWithdrawalas string with value of DebitWithdrawalDate.getText()

DECLARE pinNumber int with value of DebitPinNumbertextfield.getText()

FOR each card in cardList, do the following:

CAST the object card to a DebitCard and assign it to debitCard

IF the card's cardId is equal to cardId, then do the following:

DO

IF debitCard's withdraw method returns true when passed withdrawalAmount, dateOfWithdrawal, and pinNumber, then do the following:

DO

DISPLAY cardID and withdrawal amount in dialogue message

END DO

ELSE

DO

DISPLAY Withdrawal failed. Please check your PIN number and account balance on dialogue box

END DO

END DO

END DO

CATCH

DO

DISPLAY PLEASE ENTER CORRECT VALUES on dialogue box

END DO END

DO

END DO

IF e.getSource() is DebitDisplay **DO**

TRY DO

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

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

DECLARE bankAccount string with value of DBankAccounttextfield.getText()

DECLARE dateOfWithdrawal string with value of DebitWithdrawalDate.getText()

DECLARE cardId int with value of DCardIDtextfield.getText()

DECLARE balanceAmount int with value of DBalanceAmounttextfield.getText()

DECLARE pinNumber int with value of DPinNumbertextfield.getText()

DECLARE cardID int with value of DebitCardIDtextfield.getText()

DECLARE pinNUMBER int with value of DebitPinNumbertextfield.getText()

DECLARE withdrawalAmount int with value of debitWithdrawalAmount.getText()

CREATE a string named DEBIT_Values that concatenates the following values with line breaks in between: "Card Id:", the value of cardId, "Client Name:", and the value of clientName

FOR each debcard in cardList, do the following:

DO

IF debcard is an instance of DebitCard, then do the following:

DO

CAST debcard to a DebitCard and assign it to DebitCard_Object

INVOKE the display method on DebitCard_Object

DISPLAY Debit Card information's for setting the Debit Limit are displayed. Please check the display tab!!! On dialogue message

END DO

END DO

END DO

CATCH

DO

DISPLAY Please provide required information's only!!! On dialogue message

END DO

END DO

IF e.getSource() is SetCreditLimit **DO**

IF CCardIDTextfield, CreditCreditLimittextfield, or CreditGracePeriodtextfield is empty, then display a message dialog box that says "THE FIELDS ARE EMPTY"

END DO

ELSE

DO

TRY

DO

PARSE the integer value from CCardIDTextfield and assign it to cardId

PARSE the integer value from CreditCreditLimittextfield and assign it to newCreditLimit

PARSE the integer value from CreditGracePeriodtextfield and assign it to newGracePeriod

FOR each card in cardList, do the following:

DO

IF card is an instance of CreditCard, then do the following:

DO

CAST card to a CreditCard and assign it to cc

IF cc's cardId is equal to cardId, then set cc's credit limit to newCreditLimit and its grace period to newGracePeriod, and display a confirmation dialog box that says "Credit limit and grace period updated for card ID: " concatenated with the value of cardId

END DO

END DO

END DO

CATCH

DO

DISPLAY No credit card found with card ID on message display

END DO

END DO

END DO

IF e.getSource() is CreditDisplay **DO**

IF CCardIDTextfield, CCVCTextField, CIssuerBanktextfield, CBankAccounttextfield, CBalanceAmounttextfield, CClientNametextfield, CInterestRatetextfield, CExpirationDate, CreditCardIDtextfield, CreditCreditLimittextfield, or CreditGracePeriodtextfield is empty, then display a message dialog box that says "The fields are empty !!

END DO

ELSE

DO

TRY

DO

IF Display_CreditCard is an instance of CreditCard, then cast it to a CreditCard and assign it to CreditCard_Object. Display CreditCard_Object. Display a message dialog box that says "Credit Card information's for setting the Credit Limit are displayed. Please check the display tab!!!

END DO

END DO

END DO

CATCH

DO

DISPLAY Please provide required information's only!!! On message dialogue

END DO

END DO

END DO

IF e.getSource() is CancelCreditCard **DO**

ASSIGN the integer value of CreditCardIDtextfield.getText() to the variable cardId

LOOP through the cardList ArrayList and for each BankCard object, check if it is an instance of CreditCard

IF it is, cast it to a CreditCard object and check if its cardId matches the cardId input

IF there is a match, cancel the credit card by invoking the cancelCreditCard() method on the CreditCard object, remove the BankCard object from the cardList ArrayList, display a message in an information dialog to confirm that the credit card has been canceled, and exit the loop since the credit card has been found and canceled

END DO

IF e.getSource() is ClearButton **DO**

CLEAR all text fields in the user interface

END DO

END DO

DEFINE a function called "main" that takes no arguments and returns nothing

CREATE a new object of class "BankGUI" and assign it to a variable called "obj" **EXIT** the function

END DO

4. Method Description

A method description is a set or block of instructions. The code in a method is executed only when it is called. In the class, various methods are used for adding functionalities to the buttons. (Sheldon, n.d.)

Below are the descriptions of methods used in the BankGUI Class.

4.1 BankGUI()

A method named BankGUI is called at the beginning of the BankGUI class. Data entry into the GUI's frame is done using this technique. The method returns a void value. No parameters are accepted by it.

4.2 actionPerformed(ActionEvent e)

The ActionListener interface's actionPerformed function is defined. The method performs any operation that has to be done in response to an event, and actionListener is used to listen to the event. The method returns a void value. The ActionEvent argument is linked to an object.

4.3 AddtoDebit()

By pressing this button, the Debit Card object that the user's data created is added to the Debit Card's add frame. The data given by the user is used as a parameter to generate the object. The object is then added to the array list following that.

4.4 AddtoCredit()

By pressing this button, the user-created Credit Card object is added to the Credit Card's add frame. The user's data is used as a parameter to generate the object. The object is then included in the array list after that.

4.5 DebitWithdrawal()

By using the value specified in the Debit Card withdraw field, this button invokes the withdraw method of the Debit Card class using the user-supplied parameters.

4.6 Display()

This button displays the Debit Card or Credit Card object's value that was obtained by using the display method in the Debit Card or Credit Card Class.

4.7 SetCreditLimit()

This button executes the setCreditLimit function of the Credit Card class using the userprovided parameters after reading the value entered into the Credit Card set frame.

4.8 CancelCreditCard()

This button calls the cancelCreditCard method of the Credit Card class using the userprovided parameters after accepting the value entered into the Credit Card cancel frame.

4.9 Clear()

This button clears the fields in the GUI that have been filled in by the user. The text boxes are empty once more after being cleared.

4.10 public static void main()

The item can be accessed using this BankGUI method. This operation doesn't have a return type. The primary approach can also be used to execute the GUI directly.

5 Testing

5.1 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:	• Open command prompt • Type is • Type java BankGUI • 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 1Test that the program can be compiled and run using the command prompt

Output:

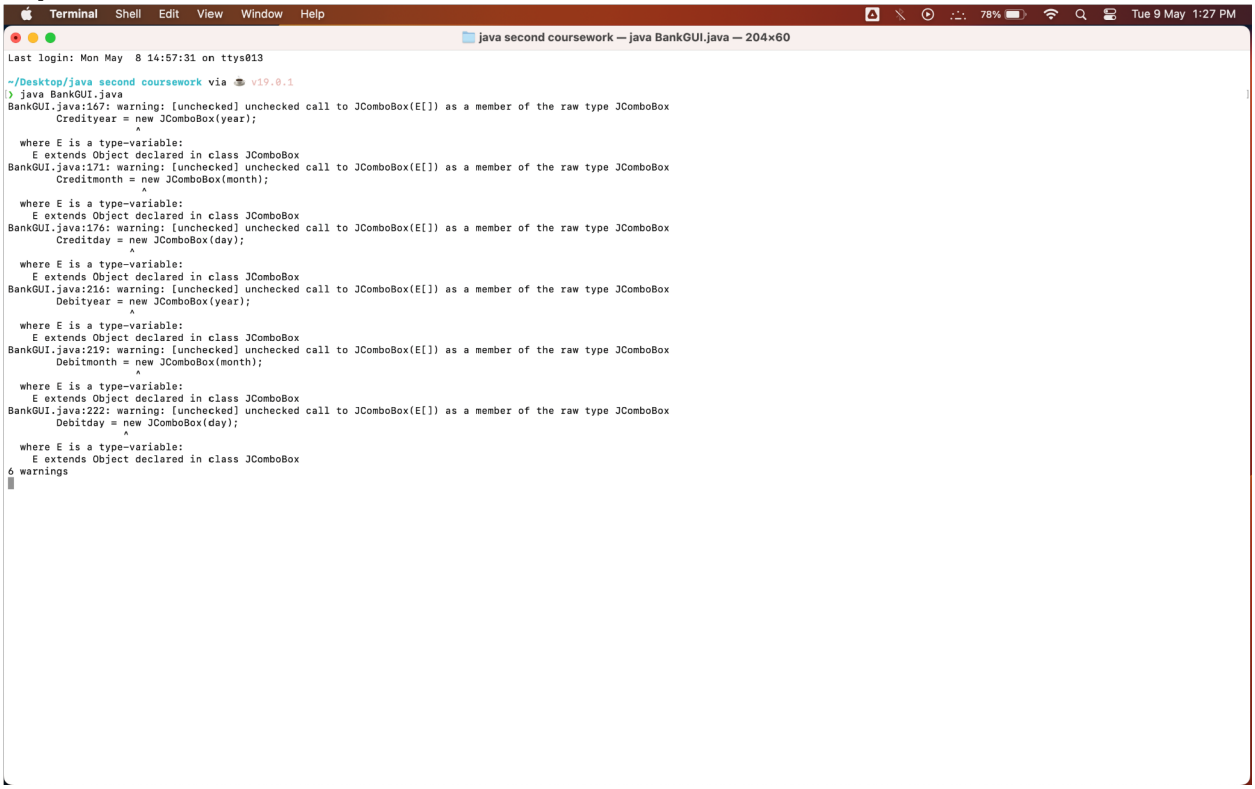


Figure 1Screenshot of class is being compiled

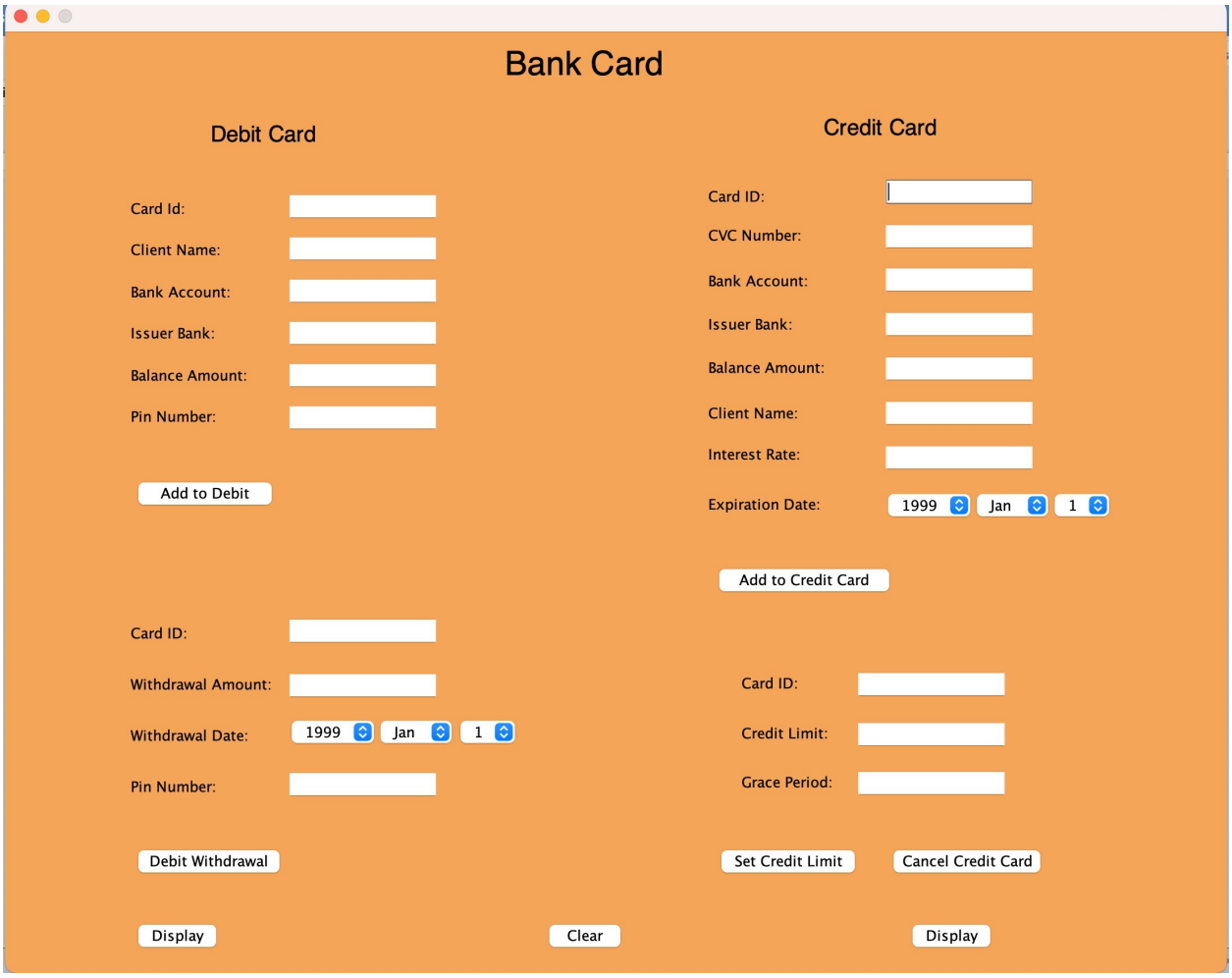


Figure 2Screenshot of BankGUI running

5.2 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 the dialogue box od added debit card.
Action:	• BankGUI is compiled and following input is given: Card Id: 123 Client Name: Bidhan Issuer Bank: Prabhu Bank Account: 123456789 Balance Amount: 100000000 Pin Number: 1212 • “Add to Debit Card” Button is pressed.
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 2 Adding Debit Card

Output:

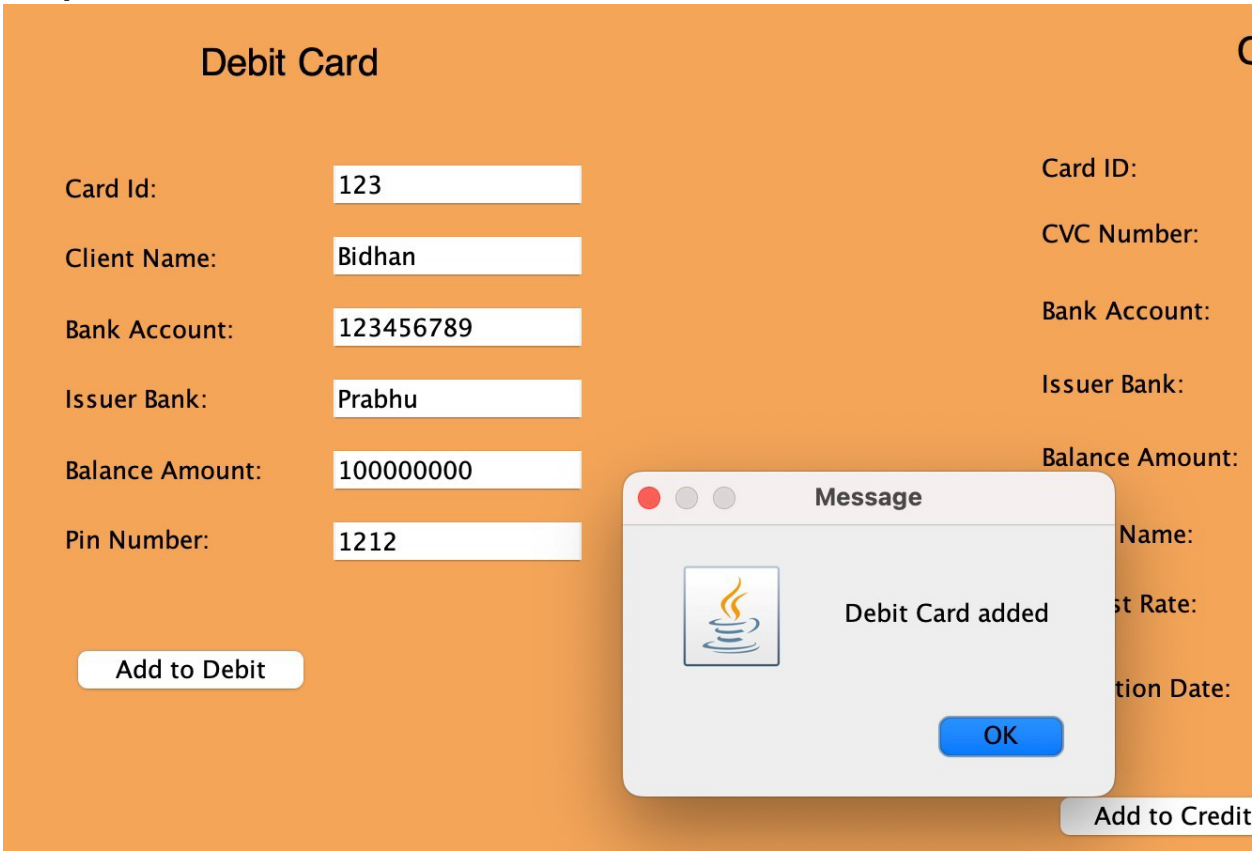


Figure 3 Screenshot of adding debit card

Test no.	2b
Objective:	To show evidences of adding Credit Card.
Action:	• BankGUI is compiled and following input is given: CardId: 123 Client Name: Bidhan Issuer Bank: Prabhu Bank Account: 123456789 Expiration Date: 2024 May 1 Balance Amount: 100000 CVC Number: 121 Interest Rate: 15 • Add to Credit Card Button
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 3 Adding of Credit Card

Output:

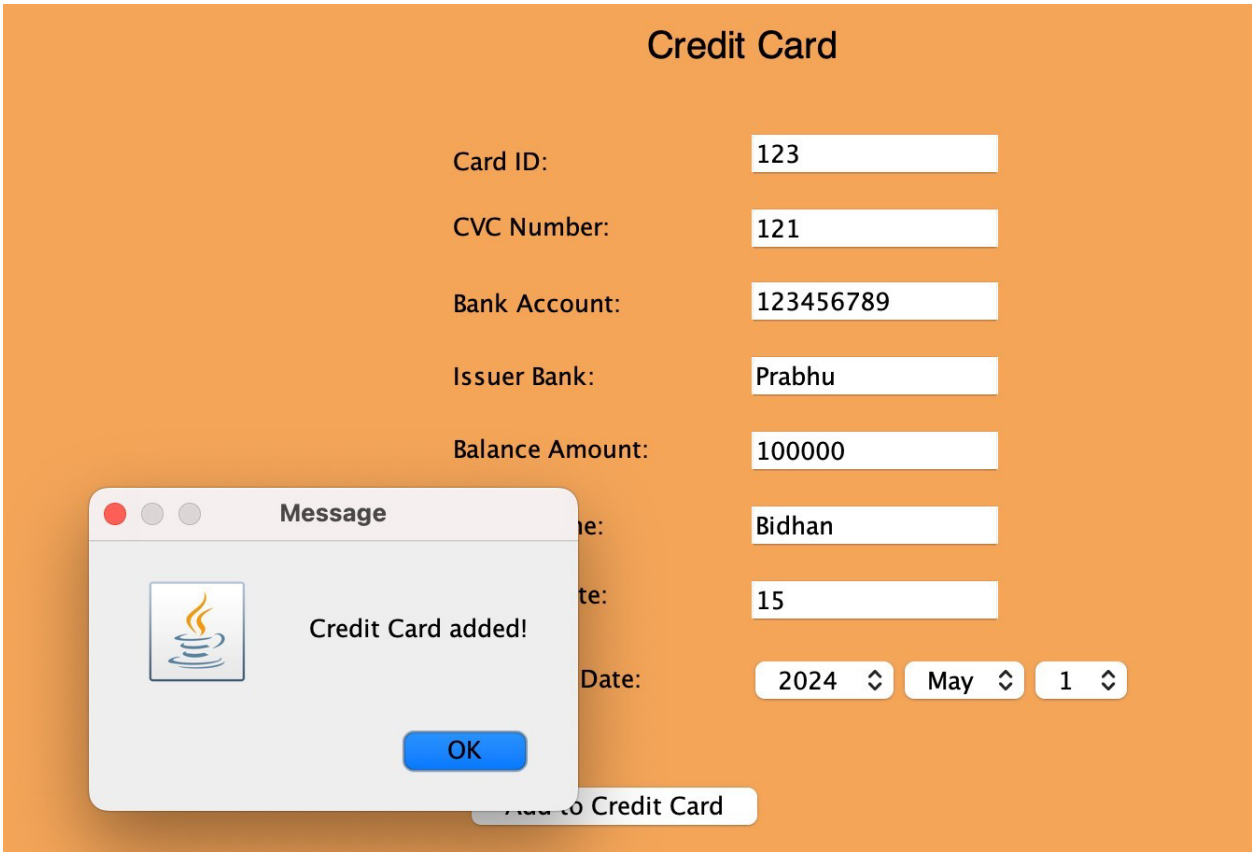


Figure 4 Screenshot of Adding Credit Card

Test no.	2c
Objective:	To show evidences of withdraw amount from Debit Card.
Action:	• BankGUI is compiled and following input is given: Card Id: 123 Client Name: Bidhan Issuer Bank: Prabhu Bank Account: 123456789 Balance Amount: 100000000 Pin Number: 1212 Card Id: 123 Withdrawal Amount: 10500 Withdrawal Date: 7 May 2023 Pin Number: 1212 • Withdraw Button is pressed.
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 4 Withdrawing from debit card

Output:

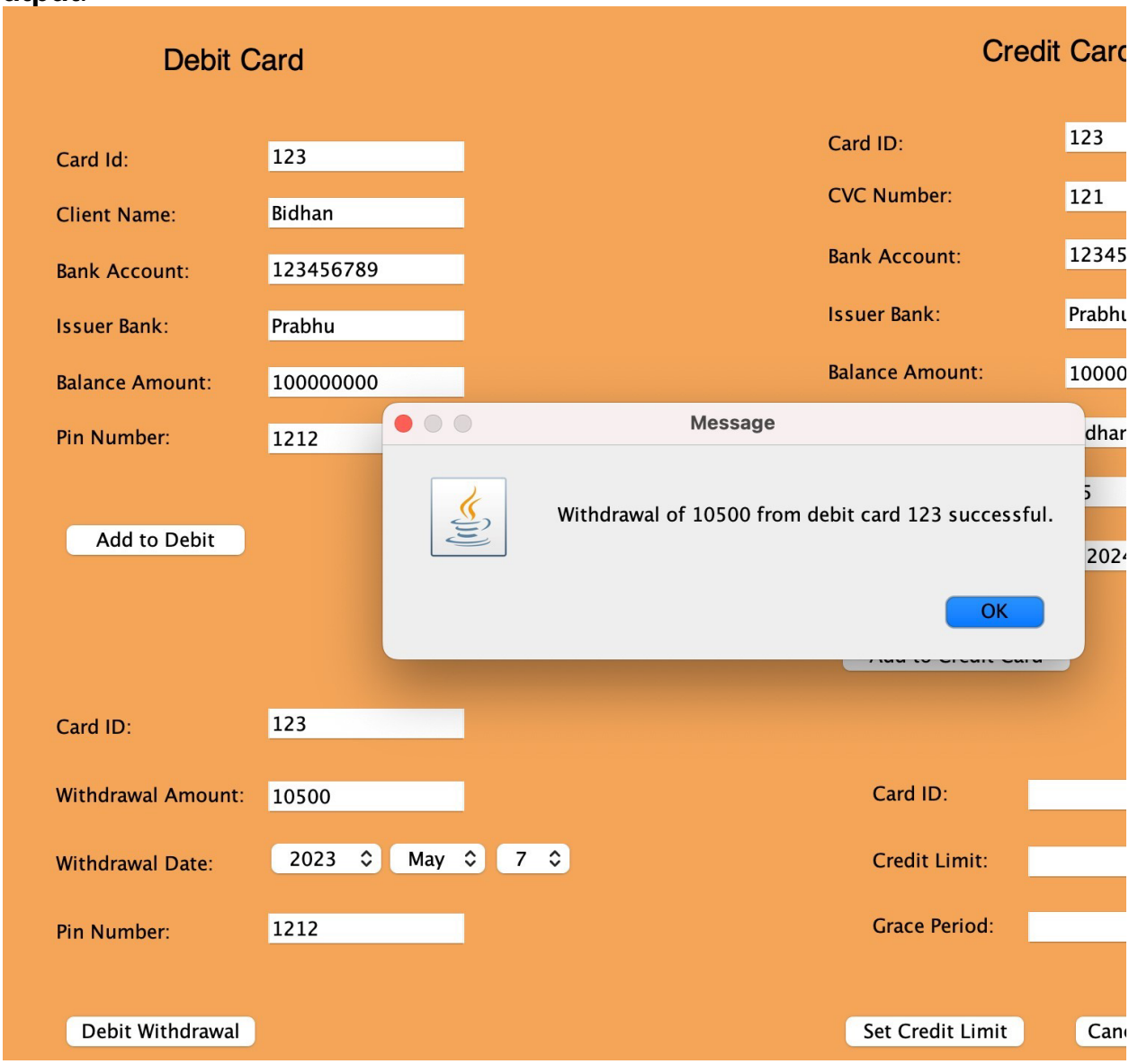


Figure 5 Screenshot of withdrawing amount from Debit Card

Test no.	2d
Objective:	To show evidences of setting credit limit.
Action:	• BankGUI is compiled and following input is given: Card Id: 123 Client Name: Bidhan Issuer Bank: Prabhu Bank Account: 123456789 Expiration Date: 2024 May 1 Balance Amount: 100000 CVC Number: 121 Interest Rate: 15 Card Id: 123 Credit Limit: 3000 Grace Period: 60 • Set Credit Limit Button is pressed.
Expected Result:	The setting of Credit Limit would be granted.
Actual Result:	The setting of Credit Limit was done.
Conclusion:	The test is successful.

Table 5 Setting Credit Limit

Output:

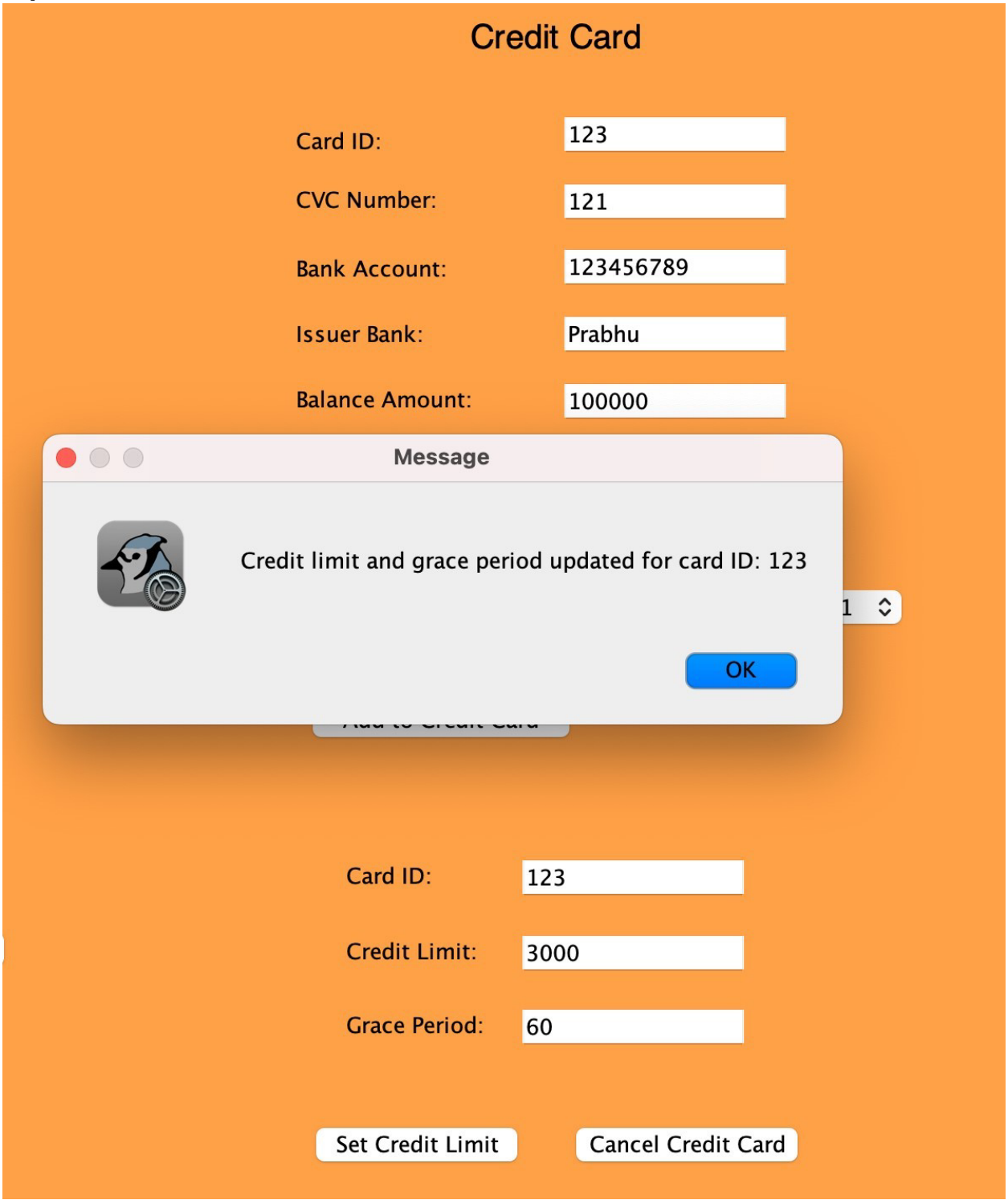


Figure 6 Screenshot of adding Credit Limit

Test no.	2e
Objective:	To show evidences of removing Credit Card.
Action:	• BankGUI is compiled and following input is given: Card Id: 123 Client Name: Bidhan Issuer Bank: Prabhu Bank Account: 123456789 Expiration Date: 2024 May 1 Balance Amount: 100000 CVC Number: 121 Interest Rate: 15 Card Id: 123 Credit Limit: 3000 Grace Period: 60 Card Id: 123 • Cancel Credit Limit Button is pressed.
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 6 Cancel Credit card

Output:

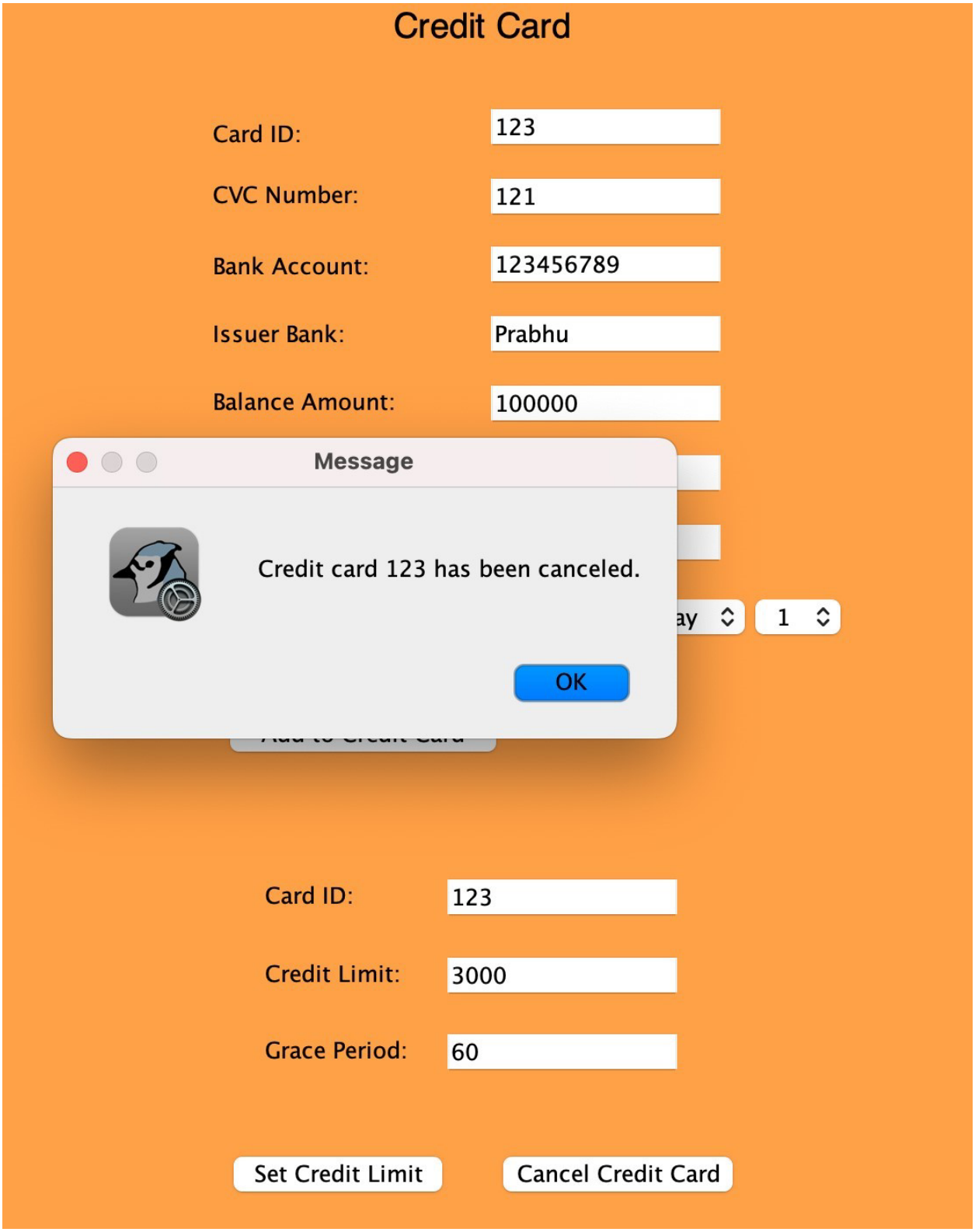


Figure 7 Screenshot of cancelling credit card

5.3 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 dialogue boxes appear when unsuitable values are entered for the Card Id.
Action:	• BankGUI is compiled and following input is given: Card Id: aa Client Name: Bidhan Issuer Bank: Prabhu Bank Account: 123456789 Balance Amount: aa Pin Number: 1212 • "Add to Debit Card" Button is pressed.
Expected Result:	The dialogue boxes would be appeared when unsuitable values are entered.
Actual Result:	The dialogue box appeared.
Conclusion:	The test is successful.

Table 7 Testing whether the dialogue box appears when unsuitable values are entered

Output:

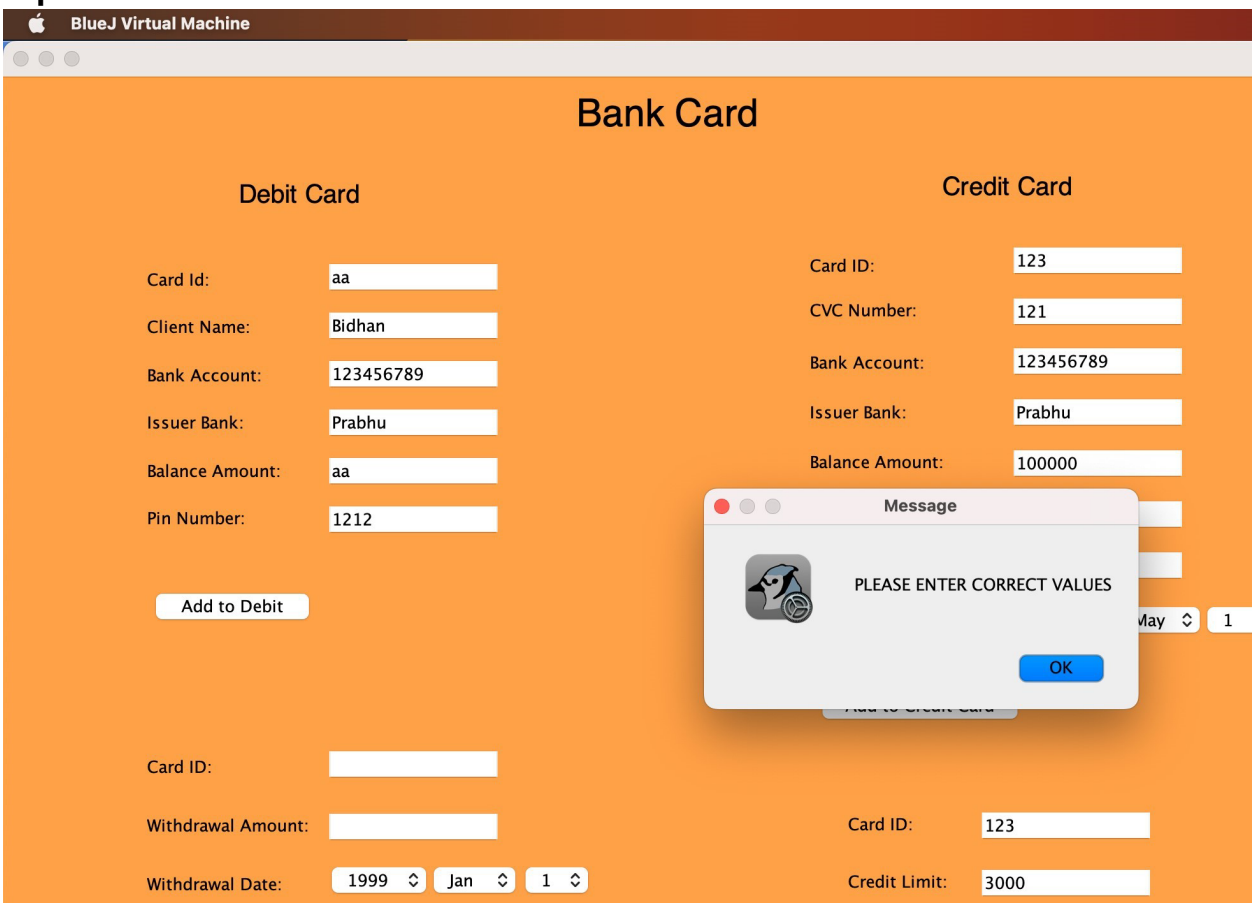


Figure 8 Screenshot of the dialogue box which appeared when unsuitable values were entered

6 Error Detection

Error detection is a technique for finding mistakes and flaws in a software; error correction is the procedure for fixing such mistakes. (Anonymous, n.d.)

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

Error 1: Syntax Error

The problems brought on when a compiler is unable to execute code because it cannot grasp the code supplied in a statement. Most syntax mistakes are omitted punctuation or misspelled names. (Attaway, 2019)

Figure 9 Syntax Error

Here in the code, there was a syntax error where a semi colon was missing.

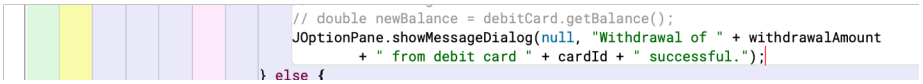


Figure 10 Syntax Error Solution

Solved the error by adding semi colon.

Error 2: Semantic Error

The term syntax refers to the formal rules that specify how a language creates true claims. Semantics is a term used to describe the set of rules that define the meaning of a statement. When programming language conventions are ignored or misused in a program, syntax problems result. lexical error. faulty program logic that results in the execution of the instructions in the wrong way. (Anonymous, n.d.)

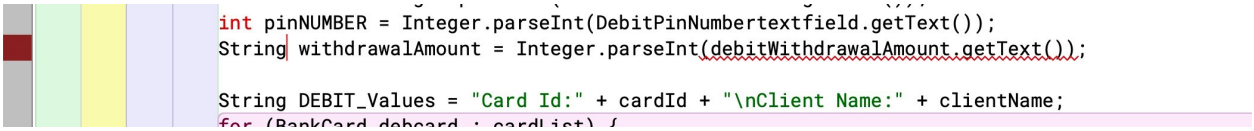


Figure 11Semantic Error

Here in the code there is a semantic error where there is wrong data type.

```
int cardID = Integer.parseInt(DebitCardIDtextfield.getText());
int pinNUMBER = Integer.parseInt(DebitPinNumbertextfield.getText());
int withdrawalAmount = Integer.parseInt(debitWithdrawalAmount.getText());

String DEBIT_Values = "Card Id:" + cardId + "\nClient Name:" + clientName;
for (BankCard debcard : cardList) {
```

Figure 12 Semantic Error Solution

Solved the error by changing the data type from string to int.

Error 3: Logical Error

Logical faults in Java programming can be difficult to detect since they don't always indicate a coding problem or an error in the use of a Java language feature. Despite not doing the task you wanted it to the code works just as intended. Therefore, logical errors could be the hardest to see.

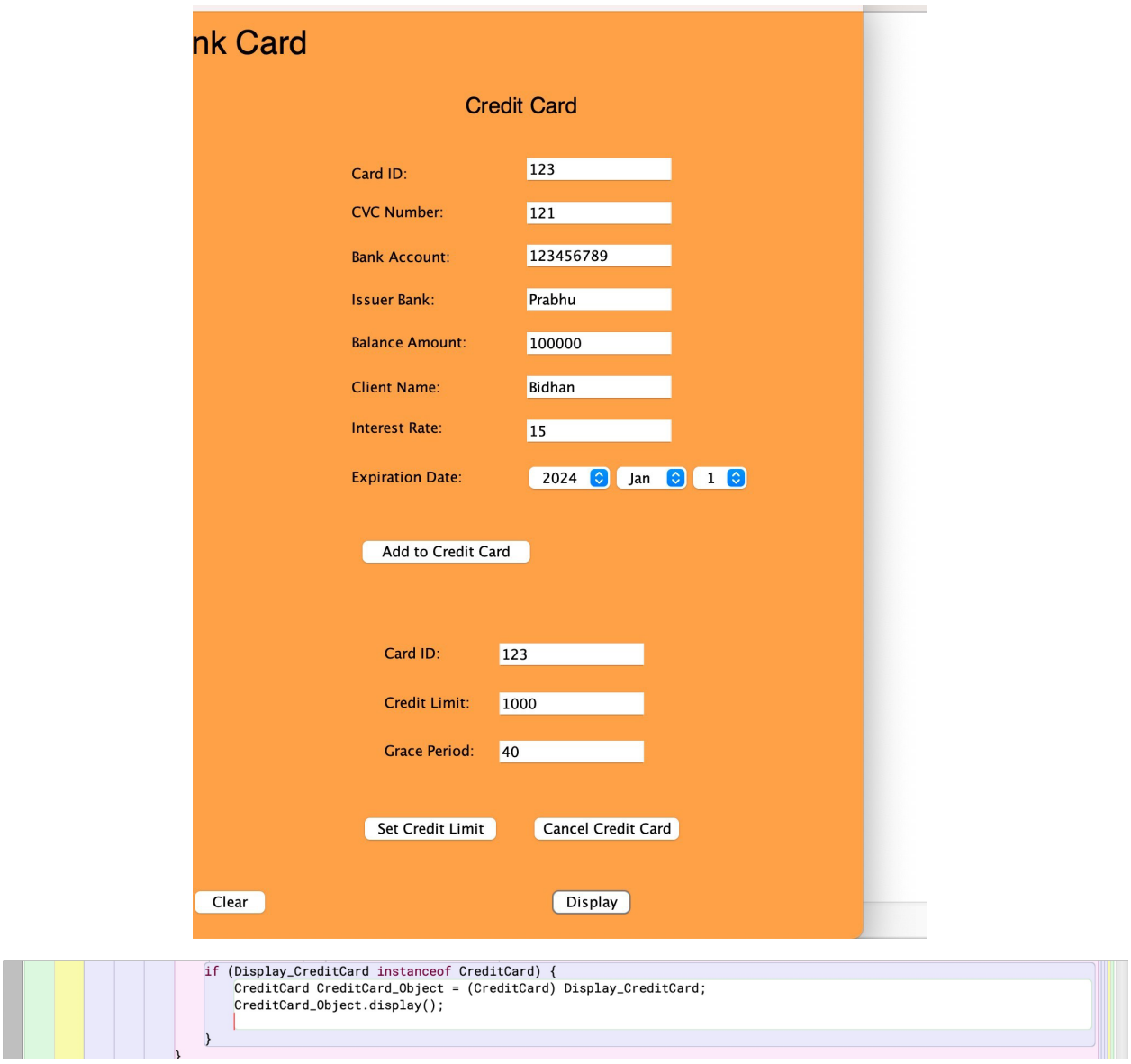
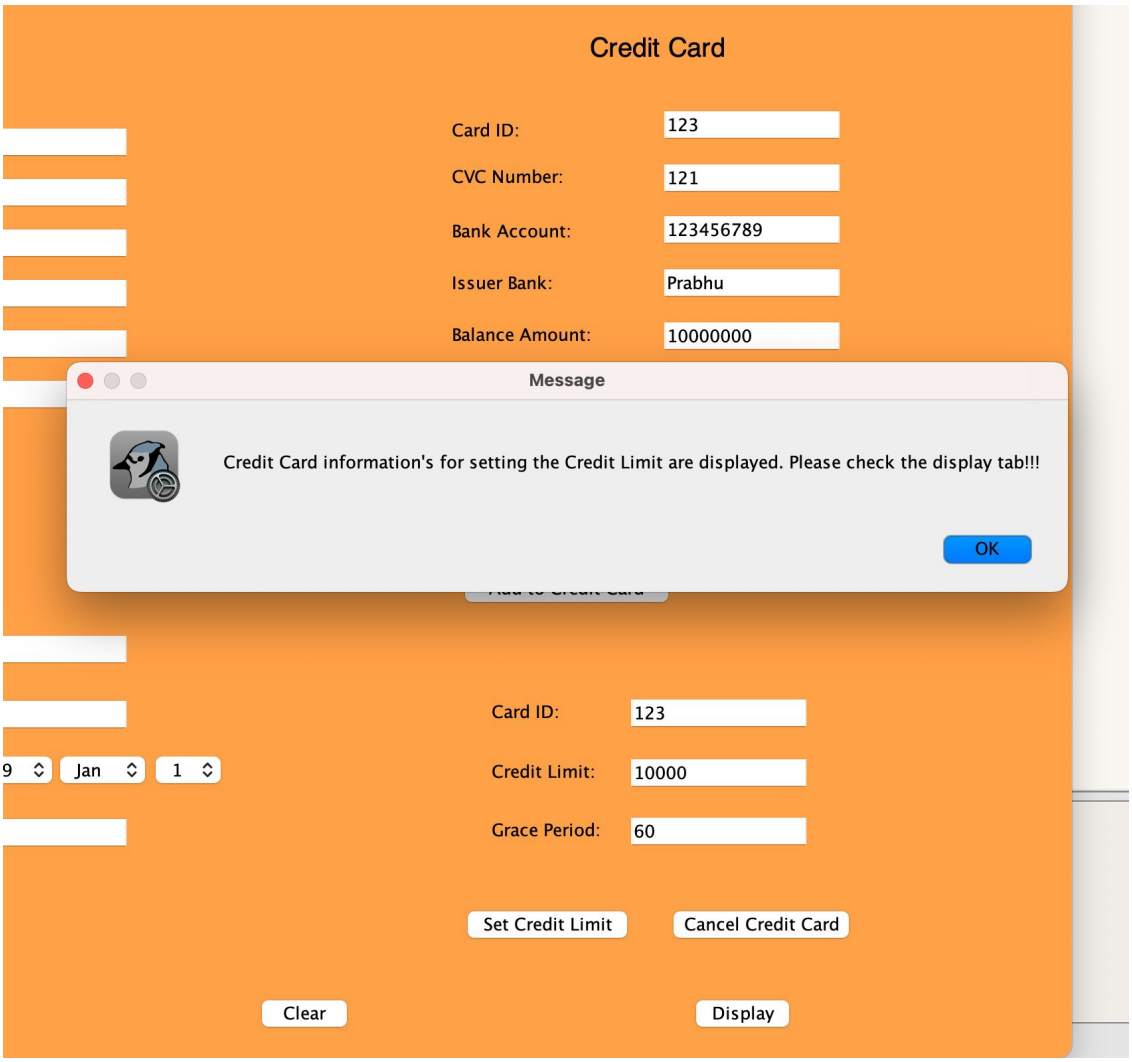


Figure 13 Logical Error

Dialogue box was not appearing on the frame.




```
CreditCard_Object.display();
JOptionPane.showMessageDialog(null,
        "Credit Card information's for setting the Credit Limit are displayed. Please check the display tab!!!");
}
```

Figure 14 Logical Error Solution

Dialogue box appeared after adding the required code.

7. Conclusion

In conclusion, this Java project explains how to use JAVA programming to construct a Graphical User Interface (GUI) for a banking system that includes the elements of a debit card and a credit card. In the BankGUI class, the array list of the BankCard class is retrieved and retained. BlueJ was used to initially write and compile the code for this Java project. The project employs many Java programming concepts, including as classes, objects, constructors, methods, looping statements, conditional expressions, inheritance, and encapsulation, in order to emulate GUI capabilities. The project also illustrates the use of the Object-Oriented Programming paradigm in the simulation's conception and execution.

Several ideas, which we learned while working on this coursework, like code authoring, pseudo code, testing, and class diagrams must be applied in order to produce an effective program. One of the key things I learned from this project was the benefits of modularity and code structure.

Following two essential project components were testing and problem-solving. Our code needs to be carefully tested to make sure it was working as planned and to catch any issues that could have occurred. The coding and implementation of the coding instructions, which were a new idea to us, were the parts of my homework that were the most difficult and offered me the greatest insight into the topic module. Overall, the experience has been both tough and gratifying.

8. References

edurika, 2023. *Edurika.* [Online]
Available at: https://www.edureka.co/blog/introduction-to-errorsinjava/#:~:text=Logical%20Errors%20%E2%80%93%20These%20are%20the,expected % 20is%20multiplication%20of%20numbers.
[Accessed 25 January 2023].

javaTpoint, 2021. *javatpoint.* [Online]
Available at: https://www.javatpoint.com/semantic-error [Accessed 25 January 2023].

Techwalla, 2023. *Techwalla.* [Online]
Available at: https://www.techwalla.com/articles/what-is-a-syntax-error-in-java
[Accessed 25 January 2023].

Draw, 2023. *Untitled Diagram.* [Online] Available at: https://app.diagrams.net/ [Accessed 25 January 2023].

Anonymous, n.d. *About BlueJ.* [Online] Available at: https://www.bluej.org/about.html [Accessed 5 May 2023].

Anonymous, n.d. *Draw.io Guide.* [Online]
Available at: https://www.coursehero.com/file/199691099/Draw-IO-Guidedocx/
[Accessed 5 May 2023].

Sheldon, R., n.d. *method(in bject oriented-programming).* [Online] Available at: https://www.techtarget.com/whatis/definition/method [Accessed 5 May 2023].

Anonymous, n.d. *Error Detection and Correction.* [Online] Available at:
https://www.tutorialspoint.com/computer_logical_organization/error_codes.htm
[Accessed 5 May 2023].

Attaway, S., n.d. *Syntax Error.* [Online]
Available at: https://www.sciencedirect.com/topics/engineering/syntax-error [Accessed 6 May 2023].

Attaway, S., 2019. *Syntax Error.* [Online]
Available at: https://www.sciencedirect.com/topics/engineering/syntax-error [Accessed 6 May 2023].

Anonymous, n.d. *Difference between syntax and semanic error.* [Online]
Available at: https://techdifferences.com/difference-between-syntax-and-semantics.html [Accessed 6 May 2023].

9. Appendix

9.1 BankGUI Class

```
import javax.swing.*; import

java.awt.*; import

java.awt.event.ActionEvent; import

java.awt.event.ActionListener; import

java.util.ArrayList;


/**

 *

* @author (22067584 Bidhan Shrestha)  * @version (1.0.0.0)

 */

public class BankGUI implements ActionListener {

   // declare all the components here

private ArrayList<BankCard> cardList;


   private JLabel label, DebitCard, CreditCard, DCardID, DClientname, DBankAccount,
DIssuerBank, DBalanceAmount,

         DPinNumber, CCardID, CCVC, CBankAccount, CIssuerBank, CBalanceAmount,
CClientName, CInterestRate,

         CExpirationDate, DebitCardID, DebitWithdrawalAmount, DebitWithdrawalDate,
DebitPinNumber, CreditCardID,

         CreditCreditLimit, CreditGracePeriod;     private JTextField DCardIDtextfield,
```

```
Dcientnametextfield, DBankAccounttextfield, DIssuerBanktextfield,
        DBalanceAmounttextfield,  DPinNumbertextfield,  CCardIDTextfield,
CCVCTextField, CBankAccounttextfield,
        CIssuerBanktextfield,  CBalanceAmounttextfield,  CClientNametextfield,
CInterestRatetextfield,
        DebitCardIDtextfield, debitWithdrawalAmount, DebitPinNumbertextfield,
CreditCardIDtextfield,
        CreditCreditLimittextfield, CreditGracePeriodtextfield;     private JButton
AddToDebit, AddtoCreditCard, DebitWithdrawal, SetCreditLimit, CancelCreditCard,
DebitDisplay,          ClearButton, CreditDisplay;     private JComboBox Crediyear,
Creditmonth, Creditday, Debityear, Debitmonth, Debitday;

   public BankGUI() {        cardList = new ArrayList<BankCard>(); //
Initialize the ArrayList


      // Jframe
      JFrame frame = new JFrame();
frame.getContentPane().setBackground(new Color(244, 165, 89));


      // main Heading  label = new JLabel("Bank
Card");        label.setBounds(450, 9, 190, 42);
label.setFont(new
Font("Helvetica", Font.PLAIN, 30));
```

```
// Debit card heading
DebitCard = new JLabel("Debit Card");
DebitCard.setBounds(185, 79, 137, 28);
DebitCard.setFont(new Font("Helvetica", Font.PLAIN, 20));

// CreditCard Heading
CreditCard = new JLabel("Credit Card");
CreditCard.setBounds(737, 73, 134, 28);
CreditCard.setFont(new Font("Helvetica", Font.PLAIN, 20));

// DebitCardid
DCardID = new JLabel("Card Id: ");
DCardID.setBounds(113, 150, 104, 19);

// Debit Client name
DClientname = new JLabel("Client Name: ");
DClientname.setBounds(113, 187, 104, 19);

// Debit Bank Account
DBankAccount = new JLabel("Bank Account: ");
DBankAccount.setBounds(113, 225, 104, 19);

// Debit Issuer Bank
```

```
        DIssuerBank = new JLabel("Issuer Bank: ");
DIssuerBank.setBounds(113, 262, 104, 19);

        // Debit Balance Amount
        DBalanceAmount = new JLabel("Balance Amount: ");
        DBalanceAmount.setBounds(113, 300, 121, 19);

        // Debit Pin Number
        DPinNumber = new JLabel("Pin Number: ");
        DPinNumber.setBounds(113, 337, 90, 19);

        // debit cardid textfield
        DCardIDtextfield = new JTextField();
        DCardIDtextfield.setBounds(253, 145, 138, 24);

        // debit client name textfield
        Dcientnametextfield = new JTextField();
        Dcientnametextfield.setBounds(253, 183, 138, 24);

        // debit bank account textfield
        DBankAccounttextfield = new JTextField();
        DBankAccounttextfield.setBounds(253, 221, 138, 24);
```

```
        // debit IssuerBank textfield
        DIssuerBanktextfield = new JTextField();
        DIssuerBanktextfield.setBounds(253, 259, 138, 24);

        // debit Balance Amount textfield
        DBalanceAmounttextfield = new JTextField();
        DBalanceAmounttextfield.setBounds(253, 297, 138, 24);

        // debit Pin Number textfield
        DPinNumbertextfield = new JTextField();  DPinNumbertextfield.setBounds(253,
335, 138, 24);

        // debit Button
        AddToDebit = new JButton("Add to Debit");
        AddToDebit.setBounds(113, 402, 134, 28);

        // credit cardID
        CCardID = new JLabel("Card ID: ");
        CCardID.setBounds(633, 140, 88, 16);

        // credit cvc
        CCVC = new JLabel("CVC Number: ");
        CCVC.setBounds(633, 175, 88, 16);
```

```
// credit bank Account

CBankAccount = new JLabel("Bank Account: ");

CBankAccount.setBounds(633, 215, 104, 19);


// credit issuer bank

CIssuerBank = new JLabel("Issuer Bank: ");

CIssuerBank.setBounds(633, 254, 98, 19);


// credit balance amount

CBalanceAmount = new JLabel("Balance Amount: ");

CBalanceAmount.setBounds(633, 293, 124, 19);


// credit client name

CClientName = new JLabel("Client Name: ");

CClientName.setBounds(633, 334, 106, 19);


// credit interest rate

CInterestRate = new JLabel("Interest Rate: "); CInterestRate.setBounds(633, 371,
107, 19);


// credit expirationdate

CExpirationDate = new JLabel("Expiration Date: ");
```

```
CExpirationDate.setBounds(633, 416, 114, 19);

// Credit cardid textfield
CCardIDTextfield = new JTextField();
CCardIDTextfield.setBounds(790, 132, 138, 24);

// Credit CVC textfield
CCVCTextField = new JTextField();
CCVCTextField.setBounds(790, 172, 138, 24);

// Credit bank account textfield
CBankAccounttextfield = new JTextField();
CBankAccounttextfield.setBounds(790, 211, 138, 24);

// Credit IssuerBank textfield
CIssuerBanktextfield = new JTextField();
CIssuerBanktextfield.setBounds(790, 251, 138, 24);

// credit Balance Amount textfield
CBalanceAmounttextfield = new JTextField();
CBalanceAmounttextfield.setBounds(790, 291, 138, 24);

// Credit Client Name textfield
```

```
CClientNametextfield = new JTextField();
CClientNametextfield.setBounds(790, 331, 138, 24);

// Credit Interest Rate
CInterestRatetextfield = new JTextField();
CInterestRatetextfield.setBounds(790, 371, 138, 24);

// Credit ComboBox
String year[] = { "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006",
"2007", "2008", "2009",
        "2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019",
"2020", "2021", "2022",
        "2023", "2024" };
Crediyear = new JComboBox(year);
Crediyear.setBounds(790, 416, 85, 23);

String month[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct",
"Nov", "Dec" };
Creditmonth = new JComboBox(month);
Creditmonth.setBounds(870, 416, 75, 23);

String day[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15",
"16", "17",
        "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30" };
```

```
Creditday = new JComboBox(day);
Creditday.setBounds(940, 416, 60, 23);

// credit button
AddtoCreditCard = new JButton("Add to Credit Card");
AddtoCreditCard.setBounds(636, 480, 167, 28);

// debit cardid
DebitCardID = new JLabel("Card ID: ");
DebitCardID.setBounds(113, 532, 60, 19);

// debit Withdrawal Amount
DebitWithdrawalAmount = new JLabel("Withdrawal Amount: ");
DebitWithdrawalAmount.setBounds(113, 578, 134, 19);

// debit Withdrawal Date
DebitWithdrawalDate = new JLabel("Withdrawal Date: ");
DebitWithdrawalDate.setBounds(113, 624, 121, 19);

// debit pin number
DebitPinNumber = new JLabel("Pin Number:  ");
DebitPinNumber.setBounds(113, 670, 104, 19);
```

```
        // debit CardID textfield
        DebitCardIDtextfield = new JTextField();
        DebitCardIDtextfield.setBounds(253, 527, 138, 24);

        // debit WithdrawalAmount textfield
debitWithdrawalAmount = new JTextField();
debitWithdrawalAmount.setBounds(253, 576, 138, 24);

        // debit pin number
        DebitPinNumbertextfield = new JTextField();
        DebitPinNumbertextfield.setBounds(253, 665, 138, 24);

        // debit button
        DebitWithdrawal = new JButton("Debit Withdrawal");
        DebitWithdrawal.setBounds(113, 733, 141, 28);

        // debit ComboBox
        Debityear = new JComboBox(year); Debityear.setBounds(253, 620, 85, 23);

        Debitmonth = new JComboBox(month);
        Debitmonth.setBounds(333, 620, 75, 23);

        Debitday = new JComboBox(day);
```

```
Debitday.setBounds(405, 620, 60, 23);


// credit cardid

CreditCardID = new JLabel("Card ID: ");

CreditCardID.setBounds(663, 577, 109, 19);


// credit credit limit

CreditCreditLimit = new JLabel("Credit Limit: ");  CreditCreditLimit.setBounds(663,
622, 109, 19);


// credit grace period

CreditGracePeriod = new JLabel("Grace Period: ");

CreditGracePeriod.setBounds(663, 666, 109, 19);


// credit cardid textfield

CreditCardIDtextfield = new JTextField();

CreditCardIDtextfield.setBounds(765, 575, 138, 24);


// credit creditlimit textfield

CreditCreditLimittextfield = new JTextField();

CreditCreditLimittextfield.setBounds(765, 620, 138, 24);


// credit graceperiod textfield
```

```
CreditGracePeriodtextfield = new JTextField();

CreditGracePeriodtextfield.setBounds(765, 664, 138, 24);


// credit buttons

SetCreditLimit = new JButton("Set Credit Limit");

SetCreditLimit.setBounds(638, 733, 134, 28);


CancelCreditCard = new JButton("Cancel Credit Card");
CancelCreditCard.setBounds(793, 733, 146, 28);


// main button

DebitDisplay = new JButton("Display");

DebitDisplay.setBounds(113, 800, 84, 28);


ClearButton = new JButton("Clear");

ClearButton.setBounds(483, 800, 78, 28);

CreditDisplay = new JButton("Display");

CreditDisplay.setBounds(810, 800, 84, 28);


// adding items to frame

frame.add(label); frame.add(DebitCard);

frame.add(CreditCard);

frame.add(DCardID);
```

```
frame.add(DClientname);
frame.add(DBankAccount);
frame.add(DIssuerBank);
frame.add(DBalanceAmount);
frame.add(DPinNumber);
frame.add(DCardIDtextfield);
frame.add(Dcientnametextfield);
frame.add(DBankAccounttextfield);
frame.add(DIssuerBanktextfield);
frame.add(DBalanceAmounttextfield);
frame.add(DPinNumbertextfield);
frame.add(AddToDebit);

        // CreditCArd frame.add(CCardID);
        frame.add(CBankAccount);
        frame.add(CCVC);
        frame.add(CBalanceAmount);
        frame.add(CIssuerBank);
        frame.add(CClientName);
        frame.add(CInterestRate);
        frame.add(CExpirationDate);
        frame.add(CCardIDTextfield);
```

```
        frame.add(CCVCTextField);
        frame.add(CBankAccounttextfield)
        ;
        frame.add(CIssuerBanktextfield);        frame.add(CBalanceAmounttextfie
        ld);
        frame.add(CClientNametextfield);        frame.add(CInterestRatetextfield);
        frame.add(AddtoCreditCard);        frame.add(Crediyear);
        frame.add(Creditmonth);        frame.add(Creditday);

        // Debit
frame.add(DebitCardID);
frame.add(DebitWithdrawalAmount);
frame.add(DebitWithdrawalDate);
frame.add(DebitPinNumber);
frame.add(DebitCardIDtextfield);
frame.add(debitWithdrawalAmount);
frame.add(DebitPinNumbertextfield);
frame.add(DebitWithdrawal);
frame.add(Debityear);
frame.add(Debitmonth);
frame.add(Debitday);
```

```
        // CreditLimit
frame.add(CreditCardID);
frame.add(CreditCreditLimit);
frame.add(CreditGracePeriod);
frame.add(CreditCardIDtextfield);
frame.add(CreditCreditLimittextfield);
frame.add(CreditGracePeriodtextfield);
frame.add(SetCreditLimit);
frame.add(CancelCreditCard);

        // main button
frame.add(DebitDisplay);

        frame.add(ClearButton); frame.add(CreditDisplay);

        ClearButton.addActionListener(this);
        AddToDebit.addActionListener(this);
        AddtoCreditCard.addActionListener(this);
        DebitWithdrawal.addActionListener(this);
        DebitDisplay.addActionListener(this);
        CreditDisplay.addActionListener(this);
        SetCreditLimit.addActionListener(this);
        CancelCreditCard.addActionListener(this);
```

```
        frame.setSize(1100,                    900);
frame.setLayout(null);        frame.setVisible(true);
frame.setResizable(false);
frame.setDefaultCloseOperation(frame.EXIT_ON
_CLOSE);

    }


    public void actionPerformed(ActionEvent e) {


        if (e.getSource().equals(AddToDebit)) {
            if       (DBalanceAmounttextfield.getText().isEmpty()                    ||
DCardIDtextfield.getText().isEmpty()
                    || DBankAccounttextfield.getText().isEmpty() ||
                    DIssuerBanktextfield.getText().isEmpty()                        ||
Dcientnametextfield.getText().isEmpty()
                    || DPinNumbertextfield.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "THE FIELDS ARE EMPTY");
            } else {
try {
                    // Get the input values                int balance =
Integer.parseInt(DBalanceAmounttextfield.getText());              int cardId =
Integer.parseInt(DCardIDtextfield.getText());
                    String bankAccount = DBankAccounttextfield.getText();
```

```
                String issuerBank = DIssuerBanktextfield.getText();
String clientName = Dcientnametextfield.getText();                  int pin
= Integer.parseInt(DPinNumbertextfield.getText());

                // Create a new DebitCard object and add it to the ArrayList
                DebitCard newCard = new DebitCard(balance, cardId, bankAccount,
issuerBank, clientName, pin);                  cardList.add(newCard);
                JOptionPane.showMessageDialog(null, "Debit Card added");
            } catch (NumberFormatException nfe) {
JOptionPane.showMessageDialog(null, "PLEASE ENTER CORRECT VALUES");
            }
        }
    }

    if (e.getSource().equals(AddtoCreditCard)) {          if
(CCardIDTextfield.getText().isEmpty() || CCVCTextField.getText().isEmpty()
                || CIssuerBanktextfield.getText().isEmpty() ||
                CBankAccounttextfield.getText().isEmpty()                          ||
CBalanceAmounttextfield.getText().isEmpty()
                || CBalanceAmounttextfield.getText().isEmpty() ||
                CClientNametextfield.getText().isEmpty()                           ||
CExpirationDate.getText().isEmpty()
                || CInterestRatetextfield.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "THE FIELDS ARE EMPTY");
```

```
        } else {
try {
            // Get the input values                int cardId =
Integer.parseInt(CCardIDTextfield.getText());                int cvc =
Integer.parseInt(CCVCTextField.getText());                String
issuerBank = CIssuerBanktextfield.getText();
            String bankAccount = CBankAccounttextfield.getText();
int balance = Integer.parseInt(CBalanceAmounttextfield.getText());
            String clientName = CClientNametextfield.getText();
String expirationDate = CExpirationDate.getText();
            double                                interestRate                                =
Double.parseDouble(CInterestRatetextfield.getText());

            // Create a new CreditCard object and add it to the ArrayList
            CreditCard newCard = new CreditCard(cardId, clientName, issuerBank,
bankAccount, balance, cvc,
                    interestRate, expirationDate);
cardList.add(newCard);
            JOptionPane.showMessageDialog(null, "Credit Card added!");
        } catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(null, "PLEASE ENTER CORRECT
VALUES");
            }
```

```
        }
    }

    if (e.getSource().equals(DebitWithdrawal)) {
if (DCardIDtextfield.getText().isEmpty() ||
debitWithdrawalAmount.getText().isEmpty()
            || DebitWithdrawalDate.getText().isEmpty() ||
DebitPinNumbertextfield.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "THE FIELDS ARE EMPTY");
        } else { try { int cardId =
Integer.parseInt(DCardIDtextfield.getText()); int withdrawalAmount =
Integer.parseInt(debitWithdrawalAmount.getText());
String dateOfWithdrawal = DebitWithdrawalDate.getText(); int pinNumber =
Integer.parseInt(DebitPinNumbertextfield.getText());

            // Find the DebitCard object in the ArrayList and withdraw the amount

            for (BankCard card : cardList) {
                DebitCard debitCard = (DebitCard) card;
if (card.getCardId() == (cardId)) {
                    // Cast the BankCard object to a DebitCard object and withdraw the
specified
```

```
                    // amount                    if
(debitCard.withdraw(withdrawalAmount,  dateOfWithdrawal,
pinNumber)) {
                        // If the withdrawal is successful, update the card balance and
display a
                        // success message
                        // double newBalance = debitCard.getBalance();
                        JOptionPane.showMessageDialog(null, "Withdrawal of " +
withdrawalAmount
                                + " from debit card " + cardId + " successful.");
                    } else {
                        // If the withdrawal fails, display an error message
                        JOptionPane.showMessageDialog(null,
                                "Withdrawal failed. Please check your PIN number and
account balance.");
                    }
                }
            }
        } catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(null, "PLEASE ENTER CORRECT
VALUES");
        }
    }
```

```
        if (e.getSource() == DebitDisplay) {
            try {
                String clientName = Dcientnametextfield.getText();
                String issuerBank = DIssuerBanktextfield.getText();
                String bankAccount = DBankAccounttextfield.getText();
                String dateOfWithdrawal = DebitWithdrawalDate.getText();

                int cardId = Integer.parseInt(DCardIDtextfield.getText());                int
balanceAmount = Integer.parseInt(DBalanceAmounttextfield.getText());                int
pinNumber = Integer.parseInt(DPinNumbertextfield.getText());                int cardID =
Integer.parseInt(DebitCardIDtextfield.getText());                int pinNUMBER =
Integer.parseInt(DebitPinNumbertextfield.getText());                int withdrawalAmount =
Integer.parseInt(debitWithdrawalAmount.getText());

                String DEBIT_Values = "Card Id:" + cardId + "\nClient Name:" + clientName;
for (BankCard debcard : cardList) {                    if (debcard instanceof DebitCard) {
                    DebitCard DebitCard_Object = (DebitCard) debcard;
                    DebitCard_Object.display();
                    JOptionPane.showMessageDialog(null,
                            "Debit Card information's for setting the Debit Limit are displayed.
Please check the display tab!!!"
                                    + DEBIT_Values);
```

```
                }
            }
        } catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(null, "Please provide required information's only!!!");
        }
    }

    if (e.getSource() == SetCreditLimit) {
        if (CCardIDTextfield.getText().isEmpty() || CreditCreditLimittextfield.getText().isEmpty()
                || CreditGracePeriodtextfield.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "THE FIELDS ARE EMPTY");
        } else {
            try {
                int cardId = Integer.parseInt(CCardIDTextfield.getText());
                int newCreditLimit = Integer.parseInt((CreditCreditLimittextfield.getText()));
                int newGracePeriod = Integer.parseInt(CreditGracePeriodtextfield.getText());

                // Search for the BankCard object with the matching card ID
                for (BankCard card : cardList) {
                    if (card instanceof CreditCard) {
                        CreditCard cc = (CreditCard) card;
                        if (cc.getCardId() == (cardId)) {
```

```
cc.setCreditLimit(newCreditLimit, newGracePeriod);
                        // show confirmation dialog to the user
                        JOptionPane.showMessageDialog(null,
                                "Credit limit and grace period updated for card ID: " + cardId);
                    }
                }
            }
        } catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(null, "No credit card found with card ID:
");
        }
    }
}

if (e.getSource() == CreditDisplay) {          if
(CCardIDTextfield.getText().isEmpty() || CCVCTextField.getText().isEmpty()
            || CIssuerBanktextfield.getText().isEmpty() ||
            CBankAccounttextfield.getText().isEmpty()                    ||
CBalanceAmounttextfield.getText().isEmpty()
            || CClientNametextfield.getText().isEmpty() ||
            CInterestRatetextfield.getText().isEmpty()                   ||
CExpirationDate.getText().isEmpty()
            || CreditCardIDtextfield.getText().isEmpty() ||
            CreditCreditLimittextfield.getText().isEmpty()               ||
CreditGracePeriodtextfield.getText().isEmpty()) {
```

```
            JOptionPane.showMessageDialog(null, "The fields are empty !!");
        } else {
try {
for (BankCard Display_CreditCa
rd : cardList) {
if
(Display_CreditC
ard instanceof
CreditCard) {
                    CreditCard CreditCard_Object = (CreditCard) Display_CreditCard;
                    CreditCard_Object.display();
                    JOptionPane.showMessageDialog(null,
                        "Credit Card information's for setting the Credit Limit are
displayed. Please check the display tab!!!");
                }
            }
        } catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(null, "Please provide required
information's only!!!");
        }
    }
}
```

```
        if (e.getSource().equals(CancelCreditCard)) {
            // Retrieve user input for card ID            int cardId =
Integer.parseInt(CreditCardIDtextfield.getText());

            // Search for the BankCard object with the matching card ID
for (BankCard card : cardList) {                if (card instanceof
CreditCard) {

                    CreditCard cc = (CreditCard) card;
if (cc.getCardId() == cardId) {
                    // Cast the BankCard object to a CreditCard object and cancel the credit
card                        cc.cancelCreditCard();

                    // Remove the credit card from the cardList ArrayList
cardList.remove(card);

                    // Display a message in an information dialog to confirm that the credit
card
                    // has been canceled
                    JOptionPane.showMessageDialog(null, "Credit card " + card.getCardId()
+ " has been canceled.");

                    // Exit the loop since the credit card has been found and canceled
                    break;
```

```
            }
        }
    }
}

if (e.getSource().equals(ClearButton)) {
    DCardIDtextfield.setText("");
    Dcientnametextfield.setText("");
    DBankAccounttextfield.setText("");
    DIssuerBanktextfield.setText("");
    DBalanceAmounttextfield.setText("");
    DPinNumbertextfield.setText("");
    CCardIDTextfield.setText("");
    CCVCTextField.setText("");
    CBankAccounttextfield.setText("");
    CIssuerBanktextfield.setText("");
    CBalanceAmounttextfield.setText("");
    CClientNametextfield.setText("");
    CInterestRatetextfield.setText("");
DebitCardIDtextfield.setText("");
debitWithdrawalAmount.setText("");
    DebitPinNumbertextfield.setText("");
    CreditCardIDtextfield.setText("");
```

```
            CreditCreditLimittextfield.setText("");
            CreditGracePeriodtextfield.setText("");
        }



    }
    public static void main(String[] args) {

        BankGUI obj = new BankGUI();
    }

}
```

9.2 BankCard Class

```
/**
* Write a description of class DebitCard here.
 *
* @author (22067584 Bidhan Shrestha)
* @version (1.0.0.0)
 */
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) {
this.balanceAmount                =
balanceAmount;       this.cardId =
cardId;          this.bankAccount =
bankAccount;       this.issuerBank
= issuerBank;       this.clientName
= ""; //initialize client name to an
empty string
    }

    //accessor methods
public int getCardId() {
return cardId;
    }
```

```
    public String getClientName() {
return clientName;
    }

    public String getIssuerBank() {
return issuerBank;
    }
    public String getBankAccount() {
return bankAccount;
    }

    public double getBalanceAmount() {
return balanceAmount;
    }

    //mutator methods     public void
setClientName(String clientName) {
this.clientName = clientName;

    }

    public void setBalanceAmount(double balanceAmount) {
this.balanceAmount = balanceAmount;
```

```
    }

    //display method
public void display() {
        System.out.println("Card ID: " + cardId);
if (clientName.equals("")) {
            System.out.println("Client name has not been 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);
    }
```

```
}1.
```

9.3 DebitCard Class

```
/**
* Write a description of class DebitCard here.
 *
* @author (22067584 Bidhan Shrestha)
* @version (1.0.0)
 */
```

```
public class DebitCard extends BankCard {
   //Attributes     private int PIN;
private int withdrawalAmount;
private String dateOfWithdrawal;
private boolean hasWithdrawn;
//Constructor     public
DebitCard(int balanceAmount, int
cardId, String bankAccount, String
issuerBank, String clientName, int
PIN) {        super(balanceAmount,
cardId, bankAccount, issuerBank);
super.setClientName(clientName);
this.PIN = PIN;        hasWithdrawn
= false;
   }
   // Accessor Method     public int getPIN() { return PIN; }     public
int getWithdrawalAmount() { return withdrawalAmount; }     public
String getDateOfWithdrawal() { return dateOfWithdrawal; }     public
boolean getHasWithdrawn() { return hasWithdrawn; }

   // Mutator Method     public void setWithdrawalAmount(int withdrawalAmount) {
this.withdrawalAmount = withdrawalAmount; }
```

```
   //Withdraw Method     public boolean withdraw(int withdrawalAmount, String
dateOfWithdrawal, int PIN) {        if (this.PIN == PIN) {            if (withdrawalAmount
<= super.getBalanceAmount()) {
super.setBalanceAmount(super.getBalanceAmount() - withdrawalAmount);
this.dateOfWithdrawal = dateOfWithdrawal;                this.withdrawalAmount =
withdrawalAmount;                this.hasWithdrawn = true;                return true;
          } else {
              System.out.println("Insufficient balance");
return false;
          }
       } else {
          System.out.println("Invalid PIN");
return false;
       }
   }



   //Display method     @Override     public void display()
{        super.display();//call th display method of
superclass        System.out.println("PIN: " + getPIN());
if(hasWithdrawn == true) {
          System.out.println("Withdrawal Amount: " + getWithdrawalAmount());
System.out.println("Date of Withdrawal: " + getDateOfWithdrawal());
```

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

9.4 CreditCard Class

```
/**
* Write a description of class DebitCard here.
 *
* @author (22067584 Bidhan Shrestha)
* @version (1.0.0)
 */
public class CreditCard extends BankCard {
    //attributes     private int
CVC;     private double
creditLimit;     private double
interestRate;     private String
expirationDate;     private int
gracePeriod;     private boolean
isGranted;
```

```
    // constructor      public CreditCard(int cardId, String clientName, String
issuerBank, String bankAccount, double balanceAmount, int CVC,     double
interestRate, String expirationDate) {         super(balanceAmount, cardId,
bankAccount, issuerBank);         this.CVC = CVC;         this.interestRate =
interestRate;         this.expirationDate = expirationDate;         this.isGranted =
false;
    }

    // accessor methods     public int getCVC() { return CVC; }
public double getCreditLimit() { return creditLimit; }     public
double getInterestRate() { return interestRate; }     public
String getExpirationDate() { return expirationDate; }     public
int getGracePeriod() { return gracePeriod; }     public
boolean getIsGranted() { return isGranted; }

    //method to set credit limit and grace period     public void
setCreditLimit(double newCreditLimit, int newGracePeriod) {         if
(newCreditLimit <= 2.5 * this.getBalanceAmount()) {

            this.creditLimit = newCreditLimit;
this.gracePeriod = newGracePeriod;
this.isGranted = true;
        } else {
```

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

    // new method to cancel credit card
public void cancelCreditCard() {
this.CVC = 0;         this.creditLimit =
0;         this.gracePeriod = 0;
this.isGranted = false;
    }

    // display method
public void display() {
if (isGranted == true) {
super.display();
System.out.println("CVC: "
+ CVC);

            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("Credit has not been granted.");
```

```
        }
    }
}
```