CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Prashant Shrestha

London Met ID: 22068050

College ID: NP01CP4A220364

Group: L1C4

Assignment Due Date: Wednesday, May 10, 2023

Assignment Submission Date: Wednesday, May 10, 2023

Contents

List of Figures:

List of Tables :

1. Introduction

This course is designed to assess students foundational knowledge of the Java programming language. The aim of this assignment is to add a class to the project that you developed for the first part of the coursework to make a graphical user interface (GUI) for a system that stores details of Bank Card in an ArrayList. The class will contain the main method and will be tested using the command prompt.

Tools that are used to do the coursework are as follows:

1.1. BlueJ

Figure 1. Bluej

As Java development environment I used Bluej , BlueJ is an Integreted Developing Environment (IDE) for the Java programing Language developed mainly for educational purposes, but also suitable for small-scale software development . It runs with the help of Java Development Kit (JDK).I generated all classes and wrote a complete program in BlueJ. It detects syntax errors and tells you exactly what's wrong, as well as suggesting fixes for handful of them. (Anon., n.d.)

1.2. MS – Word

Figure 2MS-Word

Microsoft Word is the word processing component of the Microsoft Office Suite. It is used primarily to enter, edit, format, save, retrieve and print documents. It has advanced features which allow you to format and edit your files and documents in the best possible way.The report of this course work is done on ms word. It is very easy to use, user friendly, popular and widely supported program used for multi purpose. (Anon., n.d.)

1.3 Draw.io

Figure 3Draw.io

Diagrams.io is highly regarded application software for creating any type of graphical representation. Its scope extends from simple tasks like personal mind-map to very professional engineering tasks. Its ease of access, convinience of use, easy to learn made this my go to software for creating a crucial part of this report being class diagram

1. Class Diagram

1.1. Bankcard

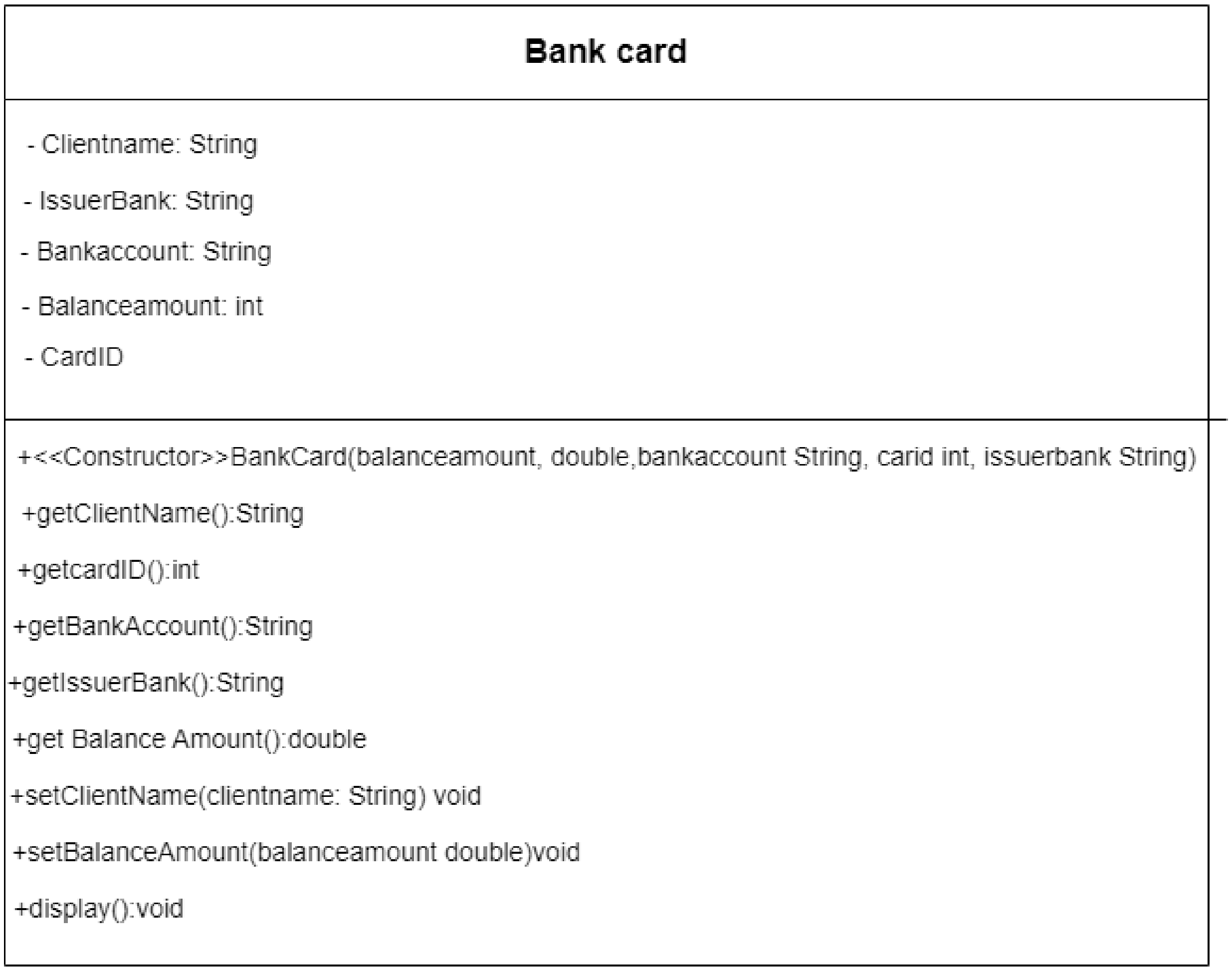

Bank card
- Clientname: String
- IssuerBank: String
- Bankaccount: String
- Balanceamount: int
- CardID
+<<Constructor>>BankCard(balanceamount, double,bankaccount String, carid int, issuerbank String)
+getClientName():String
+getcardID():int
+getBankAccount():String
+getIssuerBank():String
+get Balance Amount():double
+setClientName(clientname: String) void
+setBalanceAmount(balanceamount double)void
+display():void

Figure 4Bankcard

1.2. Debitcard

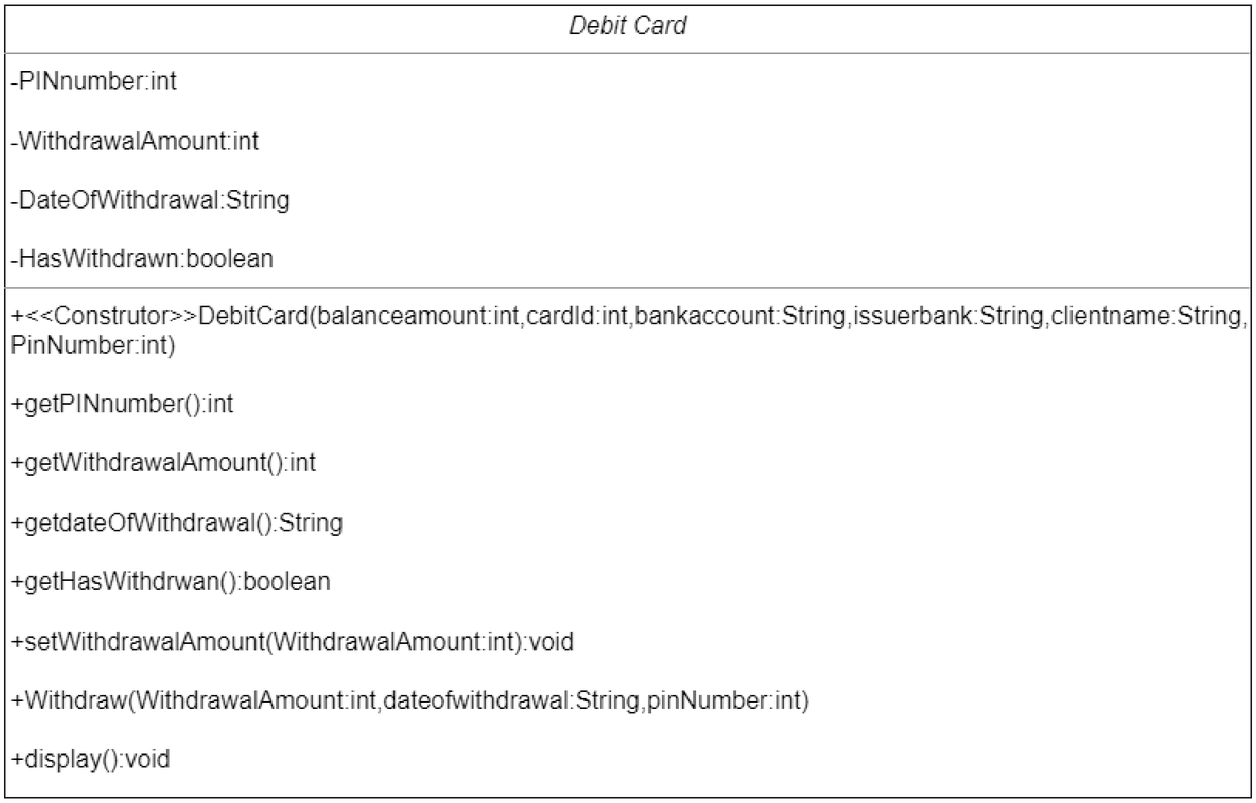
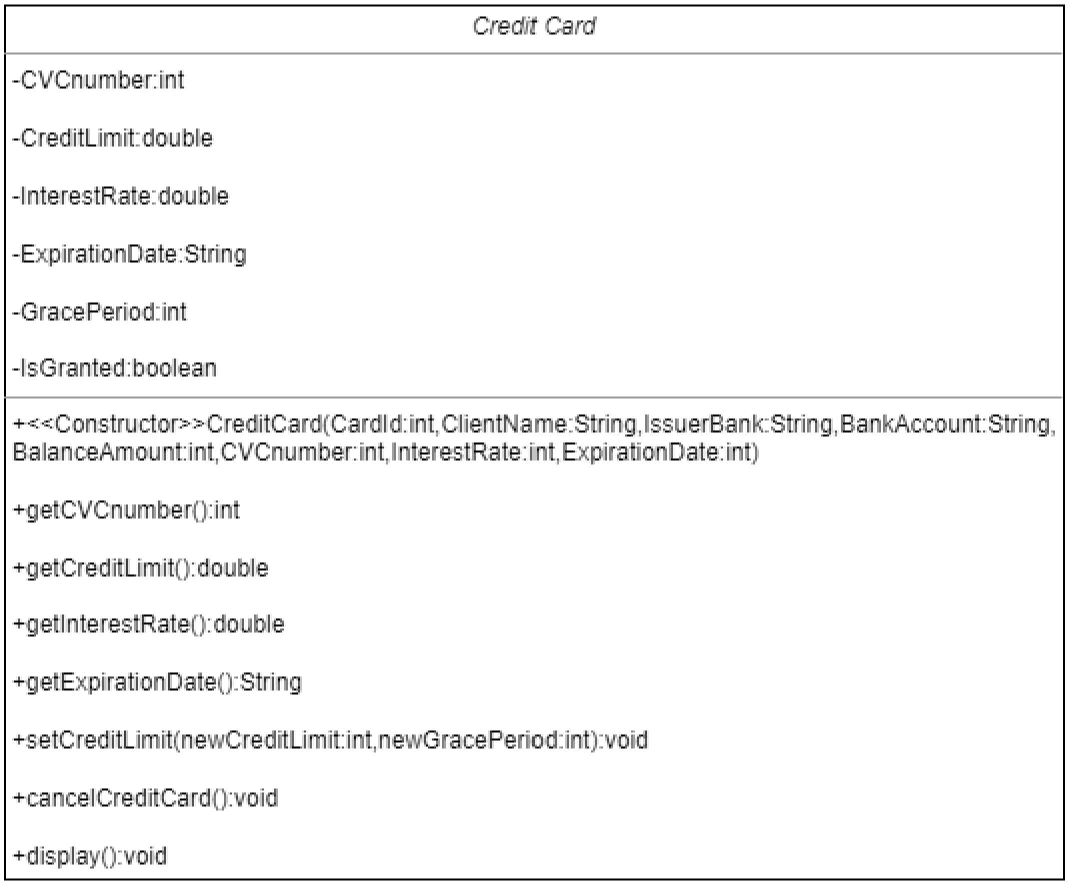

Debit Card
-PINnumber:int
-WithdrawalAmount:int
-DateOfWithdrawal:String
-HasWithdrawn:boolean
+<<Construtor>>DebitCard(balanceamount:int,cardId:int,bankaccount:String,issuerbank:String,clientname:String,PinNumber:int)
+getPINnumber():int
+getWithdrawalAmount():int
+getdateOfWithdrawal():String
+getHasWithdrwan():boolean
+setWithdrawalAmount(WithdrawalAmount:int):void
+Withdraw(WithdrawalAmount:int,dateofwithdrawal:String,pinNumber:int)
+display():void

Figure 5Debitcard

1.3. Creditcard


Credit Card

-CVCnumber:int

-CreditLimit:double

-InterestRate:double

-ExpirationDate:String

-GracePeriod:int

-IsGranted:boolean

+<<Constructor>>CreditCard(CardId:int,ClientName:String,IssuerBank:String,BankAccount:String, BalanceAmount:int,CVCnumber:int,InterestRate:int,ExpirationDate:int)

+getCVCnumber():int

+getCreditLimit():double

+getInterestRate():double

+getExpirationDate():String

+setCreditLimit(newCreditLimit:int,newGracePeriod:int):void

+cancelCreditCard():void

+display():void

Figure 6Creditcard

1.4. Combined Class Diagram

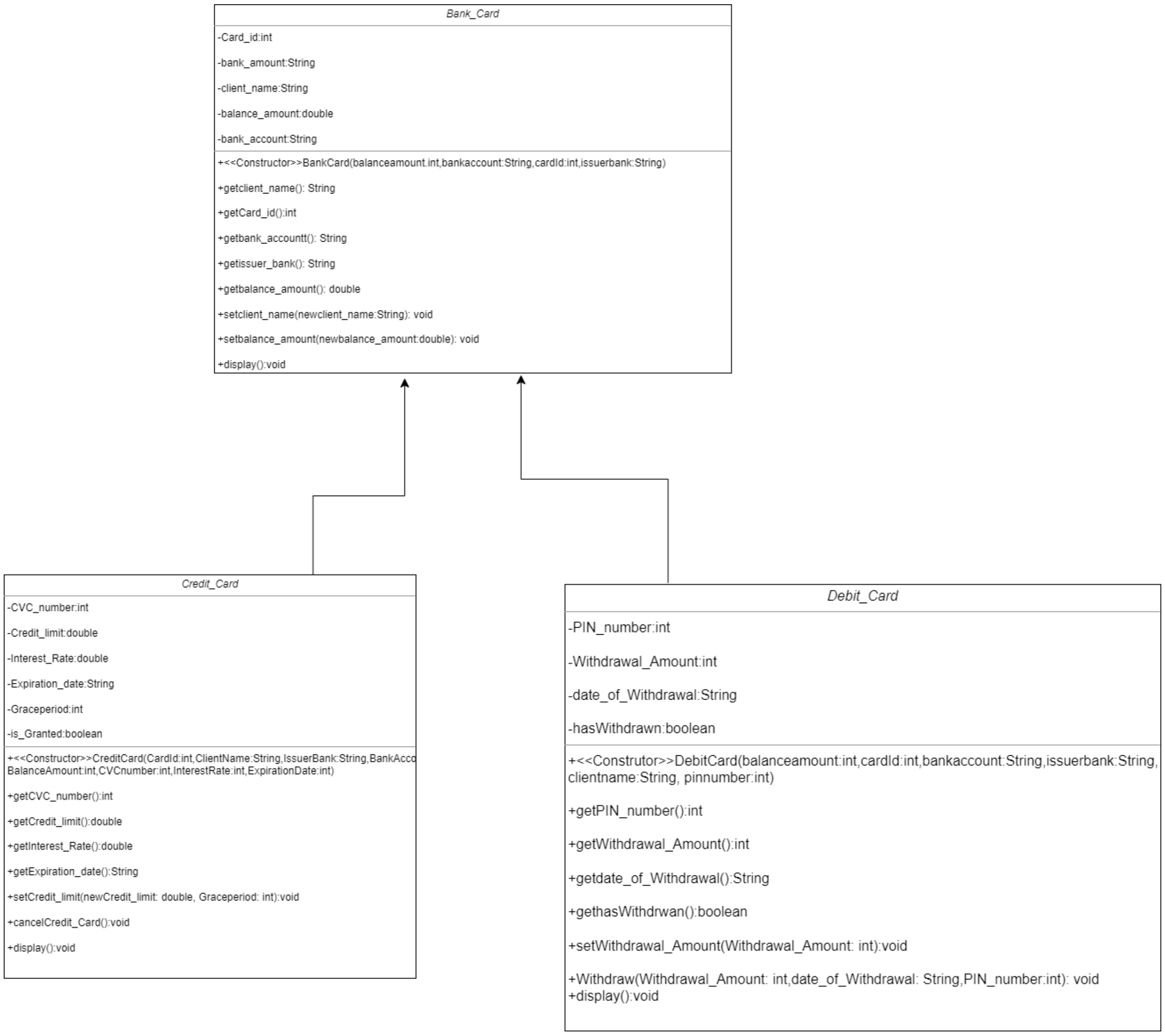
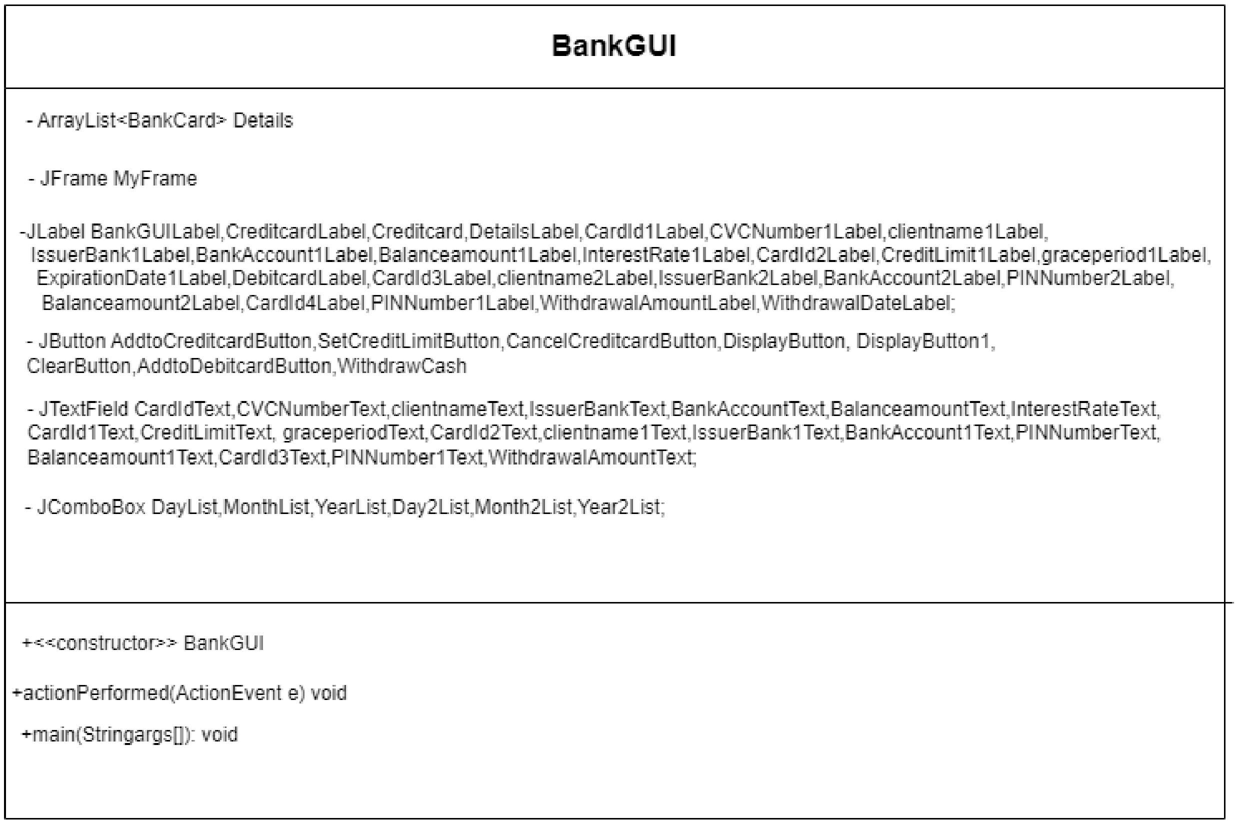

Bank_Card
-Card_id:int
-bank_amount:String
-client_name:String
-balance_amount:double
-bank_account:String
+<<Constructor>>BankCard(balanceamount.int,bankaccount:String,cardId:int,issuerbank:String)
+getclient_name(): String
+getCard_id():int
+getbank_accountt(): String
+getissuer_bank(): String
+getbalance_amount(): double
+setclient_name(newclient_name:String): void
+setbalance_amount(newbalance_amount:double): void
+display():void
Credit_Card
-CVC_number:int
-Credit_limit:double
-Interest_Rate:double
-Expiration_date:String
-Graceperiod:int
-is_Granted:boolean
+<<Constructor>>CreditCard(CardId:int,ClientName:String,IssuerBank:String,BankAcco
BalanceAmount:int,CVCnumber:int,InterestRate:int,ExpirationDate:int)
+getCVC_number():int
+getCredit_limit():double
+getInterest_Rate():double
+getExpiration_date():String
+setCredit_limit(newCredit_limit: double, Graceperiod: int):void
+cancelCredit_Card():void
+display():void
Debit_Card
-PIN_number:int
-Withdrawal_Amount:int
-date_of_Withdrawal:String
-hasWithdrawn:boolean
+<<Construtor>>DebitCard(balanceamount:int,cardId:int,bankaccount:String,issuerbank:String,
clientname:String, pinnumber:int)
+getPIN_number():int
+getWithdrawal_Amount():int
+getdate_of_Withdrawal():String
+gethasWithdrwan():boolean
+setWithdrawal_Amount(Withdrawal_Amount: int):void
+Withdraw(Withdrawal_Amount: int,date_of_Withdrawal: String,PIN_number:int): void
+display():void

Figure 7Combined Class Diagram

1.5. BankGUI Class Diagram



BankGUI

- ArrayList<BankCard> Details

- JFrame MyFrame

-JLabel BankGUILabel,CreditcardLabel,Creditcard,DetailsLabel,CardId1Label,CVCNumber1Label,clientname1Label, IssuerBank1Label,BankAccount1Label,Balanceamount1Label,InterestRate1Label,CardId2Label,CreditLimit1Label,graceperiod1Label, ExpirationDate1Label,DebitcardLabel,CardId3Label,clientname2Label,IssuerBank2Label,BankAccount2Label,PINNumber2Label, Balanceamount2Label,CardId4Label,PINNumber1Label,WithdrawalAmountLabel,WithdrawalDateLabel;

- JButton AddtoCreditcardButton,SetCreditLimitButton,CancelCreditcardButton,DisplayButton, DisplayButton1, ClearButton,AddtoDebitcardButton,WithdrawCash

- JTextField CardIdText,CVCNumberText,clientnameText,IssuerBankText,BankAccountText,BalanceamountText,InterestRateText, CardId1Text,CreditLimitText, graceperiodText,CardId2Text,clientname1Text,IssuerBank1Text,BankAccount1Text,PINNumberText, Balanceamount1Text,CardId3Text,PINNumber1Text,WithdrawalAmountText;

- JComboBox DayList,MonthList,YearList,Day2List,Month2List,Year2List;

+<<constructor>> BankGUI

+actionPerformed(ActionEvent e) void

+main(Stringargs[]): void

Figure 8 BankGUI class diagram

2. Pseudocode

Add debit card button

Check

IF the "Details" list is empty

IF the "Details" list is empty, show a message dialog indicating that the text fields are empty

IF the "Details" list is not empty, extract information from the text fields and convert them to their corresponding data types

Create a new DebitCard object using the extracted information

IFthe "Details" list is empty, add the new DebitCard object to the "Details" list and show a success message dialog

IFthe "Details" list is not empty, loop through each element of the list and check if the current card ID matches any previously added card ID

IF there is a match, show a message dialog indicating that the card has already been added

IF there is no match, add the current DebitCard object to the "Details" list and show a success message dialog

IF the user enters an invalid input format, catch a NumberFormatException and show an error message dialog

AddtoCreditcard button

IF the event source is the "AddtoCreditcardButton" button THEN

TRY to execute the following code block

GET values from the text fields

CONVERT the string inputs to their corresponding data types

CREATE a new CreditCard object using the extracted information

SET the flag variable isAdded to false

IF the "Details" list is empty THEN

ADD the credit card to the "Details" list

DISPLAY a success message

ELSE

FOR each credit card in the "Details" list DO

```
IF the current credit card ID matches any previously added credit card ID
THEN
DISPLAY a message indicating that the card has already been added
SET isAdded to false
ELSE
SET isAdded to true
END IF
END FOR
IF isAdded is true THEN
ADD the current CreditCard object to the "Details" list
DISPLAY a success message
END IF
CATCH a NumberFormatException and DISPLAY an error message if the
user inputs an invalid format
END IF
```

Display button

```
IF event source is DisplayButton1
IF Details list is empty
Display error message "Sorry, no CreditCard has been added"
ELSE
FOR each BankCard object displaycredit in Details list
IF displaycredit is an instance of CreditCard
Display success message "The details of Credit Card has been displayed"
```

```
Call the display() method for the CreditCard object
END IF
END FOR
END IF
END IF
```

SetCreditlimit button

```
IF the source of the event is SetCreditLimitButton
THEN try to:
GET the value of CardId1Text and parse it to an integer
GET the value of CreditLimitText and parse it to an integer
GET the value of graceperiodText and parse it to an integer
IF Details is empty
THEN show an error message "Cannot set Credit Limit"
ELSE iterate through the list of BankCards setCredit
IF setCredit is an instance of CreditCard AND Cardid1 is equal to setCredit's card ID
THEN set the credit limit and grace period for setCredit using the setCreditLimit method
show a success message "The Credit Limit has been set successfully"
ELSE IF Cardid1 is not equal to setCredit's card ID
THEN show an error message "The provided ID has not been found"
ELSE IF setCredit is not an instance of CreditCard
THEN show an error message "Credit Card Not Found"
CATCH a NumberFormatException and show an error message "The information you provided cannot be accepted"
```

Cancelcredit card button

GET the value from the "CardId1Text" text field and convert it to an integer called "Cardid1".

CHECK if the "Details" list is empty.

IF the "Details" list is empty, display an error message.

IF the "Details" list is not empty:

Iterate through the "Details" list using a for-each loop.

CHECK if the current BankCard in the iteration is an instance of CreditCard.

IF the current BankCard is an instance of CreditCard, check if its card ID matches the "Cardid1" integer.

IF the card ID matches, cancel the CreditCard and display a success message.

IF the card ID does not match, display an error message. If there are no CreditCards in the "Details" list, display an error message.

IF the value from the "CardId1Text" text field cannot be converted to an integer, display an error message.

Withdrawlamount button

IF event source is WithdrawCashButton THEN

TRY

GET card ID from CardId3Text

CONVERT card ID to integer CarddID1

```
GET PIN number from PINNumber1Text
CONVERT PIN number to integer pinNumber1
GET withdrawal amount from WithdrawalAmountText
CONVERT withdrawal amount to integer WithdrawalA
GET year, month and day from Year2List, Month2List, Day2List
CONCATENATE year, month and day to form dateofwithdrawal string

  IF Details is empty THEN
        DISPLAY "Cannot Withdraw, DebitCard has not been added" error
message
 ELSE
   FOR each BankCard withdrawCards in Details
   IF withdrawCards is an instance of DebitCard THEN
  IF CarddID1 is equal to withdrawCards's card ID THEN
   IF pinNumber1 is equal to withdrawCards's PIN number THEN
  IF WithdrawalA is less than or equal to withdrawCards's balance amount
THENWITHDRAW WithdrawalA amount from the withdrawCards
DebitCard with dateofwithdrawal and pinNumber1
   DISPLAY "The amount has been withdrawn successfully" success
message
  ELSE
DISPLAY "Insufficient Balance" error message
ELSE
DISPLAY "Incorrect Pin Number" error message
ELSE
 DISPLAY "DebitCard with given Id has not been found" error message
ELSE
```

```
 DISPLAY "DebitCard NOT FOUND" error message
CATCH NumberFormatException nfe
    DISPLAY "The information you provided can not be accepted" error message
END TRY
END IF
```

Displabutton1

```
IF the DisplayButton is clicked THEN
IF Details list is empty THEN
Display an error message "Sorry, no DebitCard has been added"
ELSE
FOR each BankCard displaydebit in Details list DO
IF displaydebit is an instance of DebitCard THEN
Display a success message "The details of Debit Card has been displayed"
Call the display() method of DebitCard to display its details
END IF
END FOR
END IF
END IF
```

Clear button

```
IF button "ClearButton" is clicked:
SET the text of "CardIdText" to empty
SET the text of "CVCNumberText" to empty
SET the text of "clientnameText" to empty
SET the text of "IssuerBankText" to empty
SET the text of "BankAccountText" to empty
SETthe text of "BalanceamountText" to empty
SET the text of "InterestRateText" to empty
SET the text of "CardId1Text" to empty
SET the text of "CreditLimitText" to empty
SET the text of "graceperiodText" to empty
SET the text of "CardId2Text" to empty
SET the text of "clientname1Text" to empty
SET the text of "IssuerBank1Text" to empty
SET the text of "BankAccount1Text" to empty
SET the text of "PINNumberText" to empty
SET the text of "Balanceamount1Text" to empty
SET the text of "CardId3Text" to empty
SET the text of "PINNumber1Text" to empty
SET the text of "WithdrawalAmountText" to empty
```

3. Description of methods

3.1. action Performed()

Method	Description
actionPerformed()	The ActionListener interface includes a common method called actionPerformed(). Every time, it is automatically called. A button click or menu item selection is an example of an action event. The actionPerformed() method of the associated listener will be called automatically when the user interacts with the component. public void actionPerformed(ActionEvent) is the method signature. Information about the event that called the method, such as its origin, is contained in the ActionEvent parameter. We have written the code that needs to run when the appropriate action is performed in the associated buttons inside it inside the actionPerformed() method.

Table 1 Method description of action performed

Add to Debit Card button

Method	description
get source():	It is applied here to evaluate whether the Add to Debit card button was responsible for the incident.
getText():	The text entered in a number of text fields, including ClientName1, IssuerBank1, BankAccount1, BalanceAmount1, CardID2, and PINNumber1, is used in this code.
isEmpty():	This method is used to determine whether a string is empty. Checking whether any of the mandatory text boxes are empty is done in this code.
parseDouble():	Using this method, a string can be converted into a double. The text entered in the BalanceAmount1 text field is parsed in this code.
parseInt():	With the method, an integer can be extracted from a string.
getCard_ID():	The CardID2 value of a Debit Card object can be obtained using the function. The CardID2 value of the current Debit Card object in the bank ArrayList is compared to the CardID2
new DebitCard():	a new DebitCard object is created in this code
add():	A new object can be added to an ArrayList using the add() method
showMessageDialog():	using this function. The user will see a variety of messages depending on the results of the input validation and object creation processes

Table 2 Method description table of Add to debit card button

3.2. Add to credit card button

Method	Description
get source():	It is applied here to evaluate whether the Add to Credit card button was responsible for the incident.
getText():	The text entered in a number of text fields, including ClientName, IssuerBank, BankAccount, BalanceAmount, CardID, and CVC number , Interest rate , is used in this code.
isEmpty():	This method is used to determine whether a string is empty. Checking whether any of the mandatory text boxes are empty or not
parseDouble():	Using this method, a string can be converted into a double. The text entered in the BalanceAmount text field is parsed in this code.
parseInt():	With the method, an integer can be extracted from a string.
getCard_ID():	The CardID value of a Debit Card object can be obtained using the function. The CardID value of the current Debit Card object in the bank ArrayList is compared to the CardID
new CreditCard():	a new CreditCard object is created in this code
add():	A new object can be added to an ArrayList using the add() method
showMessageDialog():	using this function. The user will see a variety of messages depending on the results of the input validation and object creation processes

Table 3 Method description add to Credit card button

3.3. Withdrawal amount Button

Method	**description**
getSource():	Using this function. The WithdrawButton button's involvement in the event is checked
getText():	The text entered into a text field can be obtained using this method. The content entered in numerous text fields, including WithdrawalAmountT, CardIDT1, and PINNumberT1, is utilised
isEmpty():	This method is used to determine whether a string is empty. Checking whether any of the mandatory text boxes are empty or not
getSelectedItem():	The selected item in a drop-down menu can be obtained using this method
parseInt():	With the method, an integer can be extracted from a string.
instanceof:	To determine whether an object is an instance of a particular class, use this keyword. It is used to determine whether the current objectinside a bank An instance of DebitCard is ArrayList.
getCard_ID():	Get the CardID value of a DebitCard object using the getCardID() function. In this code, the CardID value of the current DebitCard object in the bank ArrayList is utilized to compare with The user entered CardID

getPIN_Number():	The PINNumber value of a DebitCard object can be obtained using this function. The PINNumber value of the current DebitCard object in the bank ArrayList is compared with the PINNumber entered by the user in this code.
getBalance_Amount():	This function determines whether the withdrawal amount entered by the user is less than or equal to the Debit Card object's current balance.
showMessageDialog():	Using this function. The user will see a variety of messages depending on the results of the input validation and object creation processes
Withdraw():	This method is used to withdraw a specified amount from a DebitCard object. In this code, it is used to withdraw the amount entered by the user from the DebitCard object if the withdrawalis valid.

Table 4 Method description of with draw button

3.4. Cancel Credit Card Button

Method	description
getSource():	Get the event's source object using the getSource() function. It is applied here to assess whether the CancelB button was to blame for the incident.
getText():	The text entered into a text field can be obtained using the getText() method. The text entered in the CardIDT4 text field is utilized in this code to retrieve it.
sEmpty():	Use the isEmpty() method to determine whether a string is empty. It is in this code enables one to determine whether the CardIDT4 text field is empty.
parseInt():	With the parseInt() method, an integer can be extracted from a string. The text entered in the CardIDT4 text field is parsed in this code.
instanceof:	To determine whether an object is an instance of a specific class, use the instanceof keyword. Checking whether the current object in the bank ArrayList is an instance of CreditCard is done
cancelCreditCard():	In order to cancel a credit card, cancelCreditCard() sets its balance to $0 and disables it. The Card_ID given by the user is used to cancel a CreditCard object
new CreditCard():	a new CreditCard object is created in this code
add():	A new object can be added to an ArrayList using the add() method
showMessageDialog():	using this function. The user will see a variety of messages depending on the results of the input validation and object creation processes

Table 5 Method description of cancel credit card

3.5. Clear Button

Method	Description
getSource():	This method is used to get the source object of the event. In this code, it is used to determine if the event was caused by the Clear button.
setText():	This method is used to set the text in a text field. In this code, it is used to set the text of various text fields such as CardIdText,CVCNumberText,clientnameText,IssuerBankText,BankAccountText,BalanceamountText,InterestRateText,CardId1Text,CreditLimitText,graceperiodText,CardId2Text,clientname1Text,IssuerBank1Text,BankAccount1Text,PINNumberText,Balanceamount1Text,CardId3Text,PINNumber1Text,WithdrawalAmountText;

Table 6 Method description table of clear button

3.6. Display button

Method	**description**
getSource():	This method is used to get the source object of the event. In this code, it is used to determine if the event was caused by the Display button.
display():	The CreditCard class contains a method called display() that is used to show an object's data. The purpose of this code is to list all CreditCard objects in the bank's details. ArrayList.

Table 7Method description of display button

3.7. Display button1

Method	**description**
getSource():	This method is used to get the source object of the event. In this code, it is used to determine if the event was caused by the Display button.
display():	A DebitCard object's details are printed out using the DebitCard class's custom function display(). The purpose of this code is to list all DebitCard objects in the bank's details. ArrayList.

Table 8 Display button1 table

3.8. Method Description of main()

Method	**description**
main():	main() This is the method signature: void main (String args[]) public static { New BankGUI(); One line of code in the main() method creates a new instance of the BankGUI class. This class most likely functions as a user interface that enables users to communicate with the bank system. The software begins the user interface and enables users to start interacting with the system by establishing a new instance of this class. An array of command-line arguments can be supplied to the program when it is run using the String args[] parameter. These parameters can be used to give the program input or configuration options.

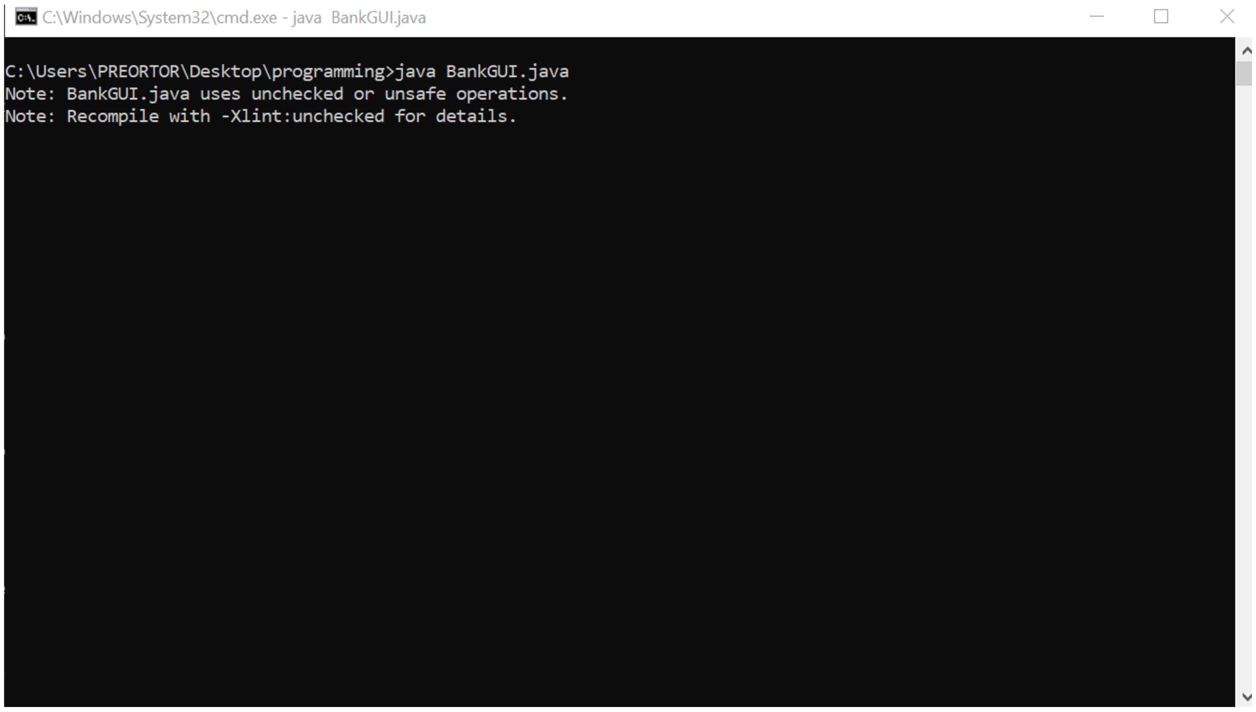
Table 9 Table if method description of main method

4. Testing

4.1. Test 1:

Test that the program can be compiled and run using the command promt, Including screen shot

i. Running the program using command prompt

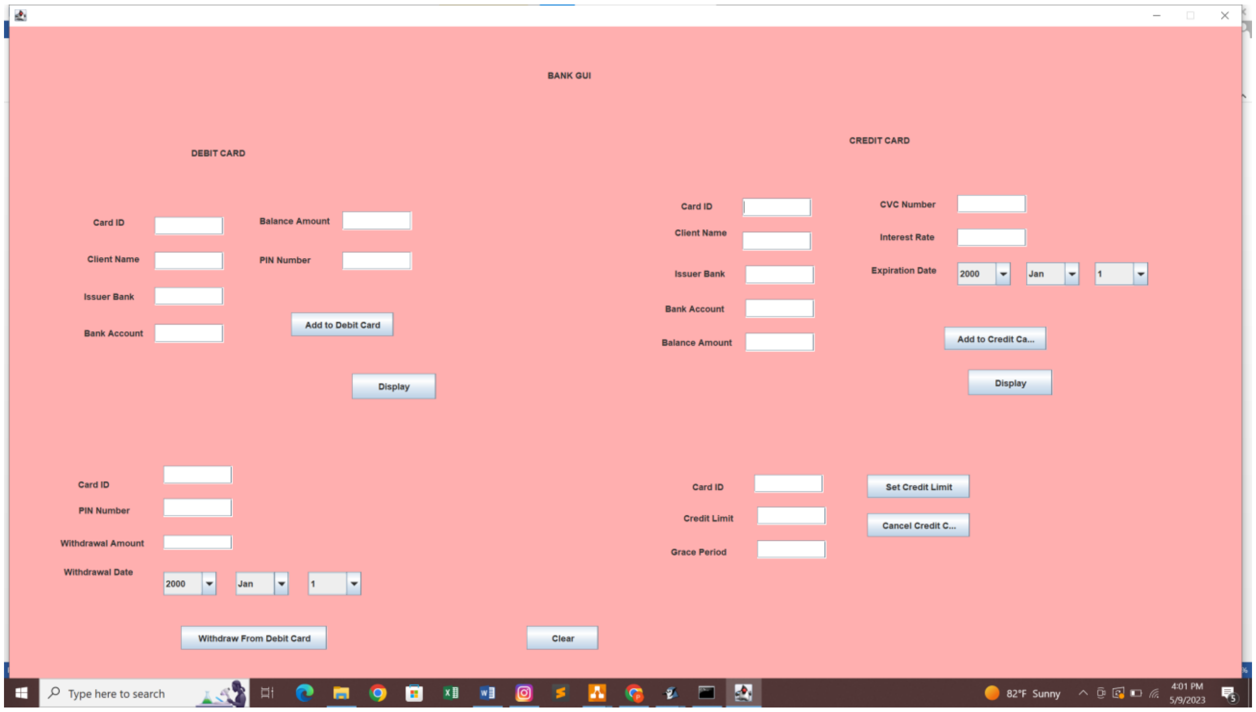


C:\Windows\System32\cmd.exe - java BankGUI.java

```
C:\Users\PREORTOR\Desktop\programming>java BankGUI.java
Note: BankGUI.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Figure 9 Screen shot of command prompt

ii. GUI opened after running program in command prompt



Objective :	Test that the program can be compiled and run using Comand prompt
Action:	➢ Command prompt is opened with the path of java package ➢ The command java BankGUI .java is used
Expected Result:	A java frame would be opened
Actual Result:	A java frame was opened
Conclusion	The test is successful.

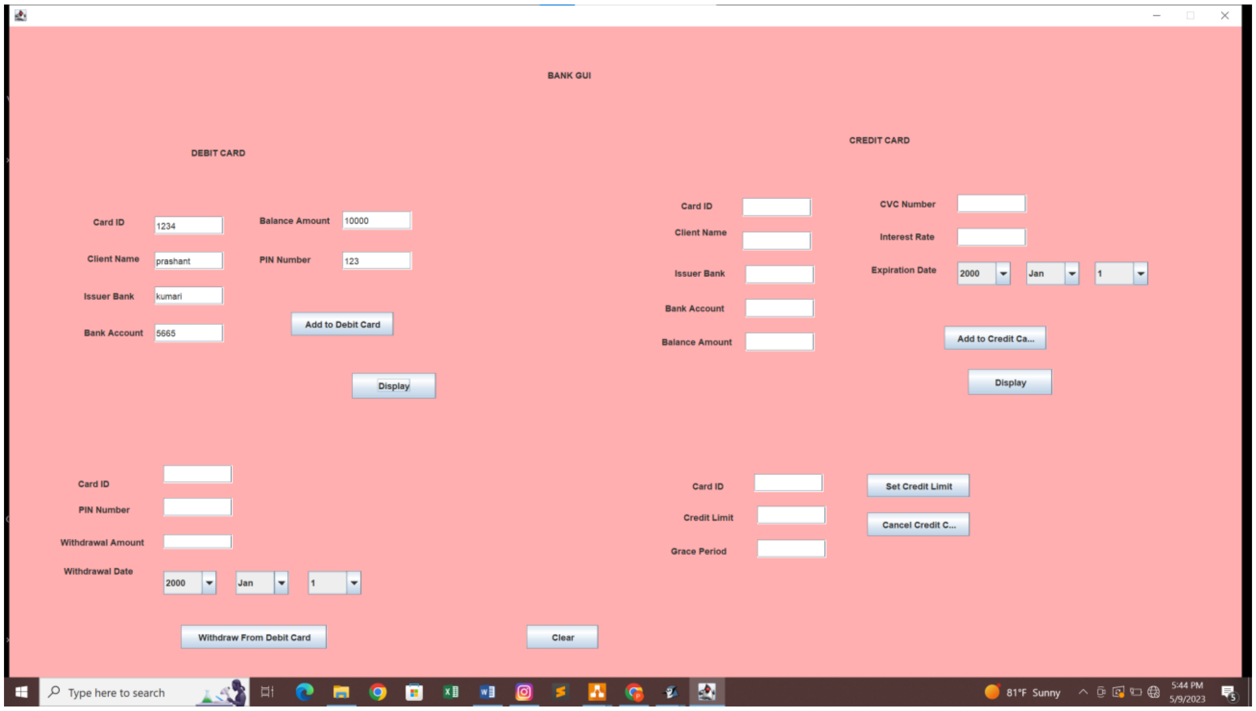
Figure 10 Screenshot of GUI

Table 10 Test table

4.2. Test 2: Evidences should be shown of:

AddtoDebitCard Button:

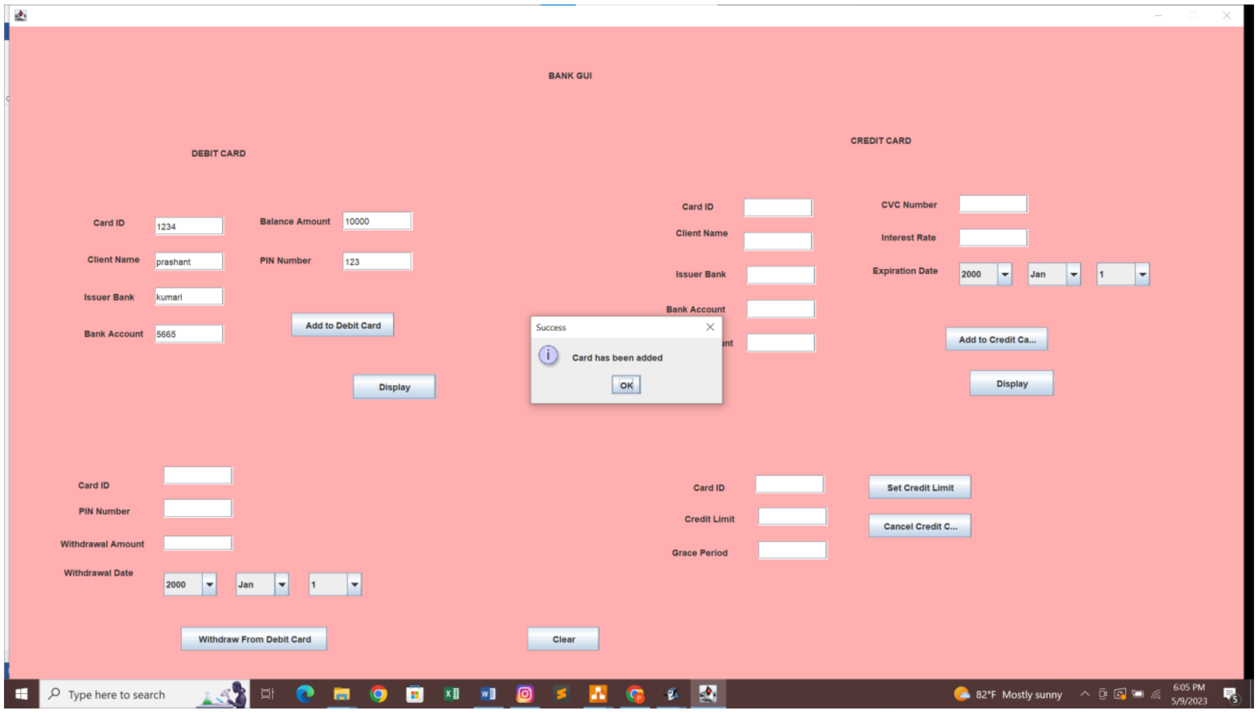
i. Adding the debit card to arraylist



Card ID: 1234
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
Balance Amount: 10000
PIN Number: 123

Figure 11 adding the Debit card to arraylist

ii. Function of button and message after add to debit card button is pressed



Card ID 1234
Client Name prashant
Issuer Bank kumari
Bank Account 5665
Balance Amount 10000
PIN Number 123

Success
Card has been added

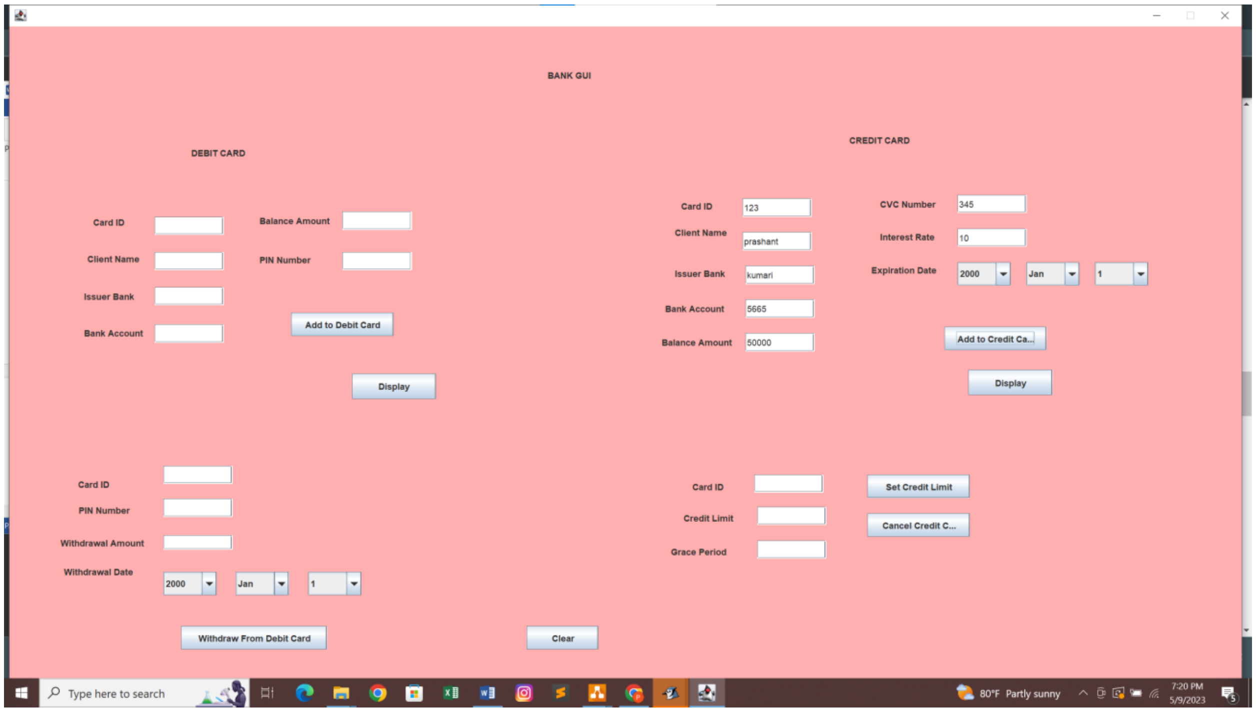
Figure 12Screenshot of displayed message

Objective :	To input values of balance amount, card Id, bank account, issuer bank, client name, PIN number in arraylist
Action:	➢ Input values in required field ➢ Press the button
***Expected Result:**	Card has been added
Actual Result:	Inputted values are add to the arraylist
Conclusion:	The test is successful.

Table 11 Test table AddtoDebitCard button

AddtoCreditCard Button

i. Adding the Credit card to arraylist

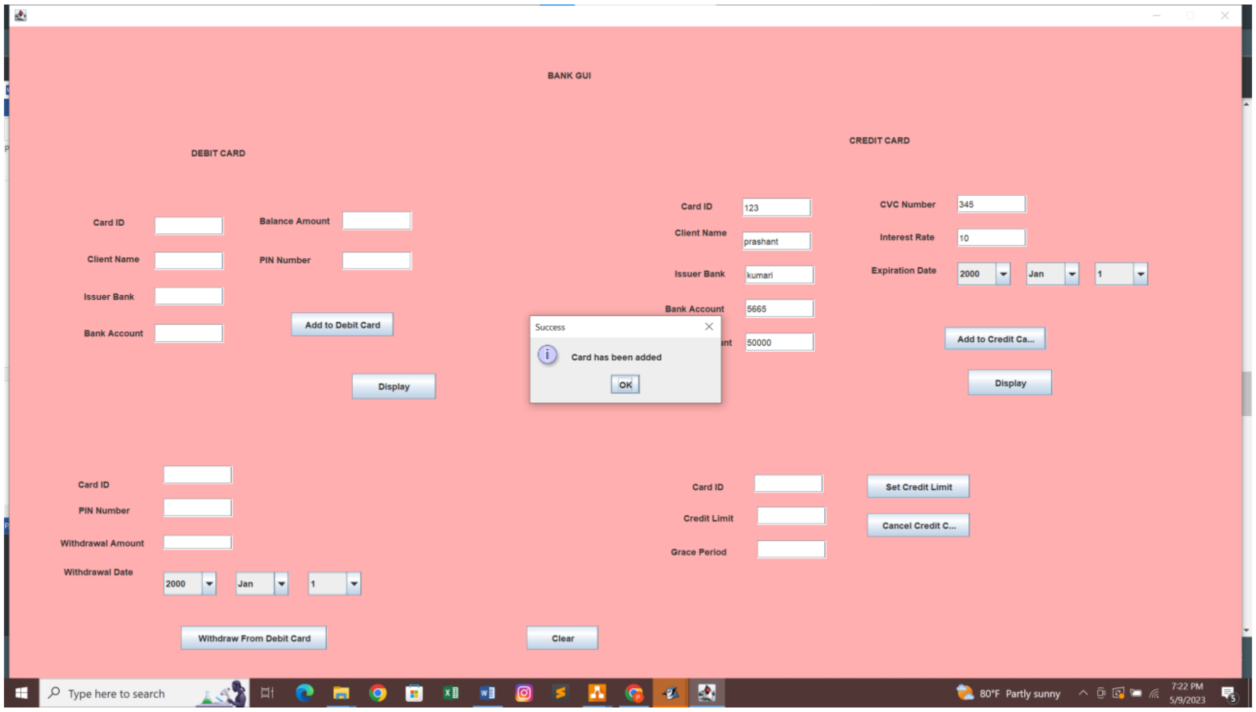


Withdrawal Date: 2000 Jan 1

Card ID: 123
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
Balance Amount: 50000
CVC Number: 345
Interest Rate: 10
Expiration Date: 2000 Jan 1

Figure 13 Adding Creditcard to arraylist

ii. Function of button and message after add to credit card button is pressed



Card ID: 123
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
50000
CVC Number: 345
Interest Rate: 10
Expiration Date: 2000 Jan 1

Success
Card has been added
OK

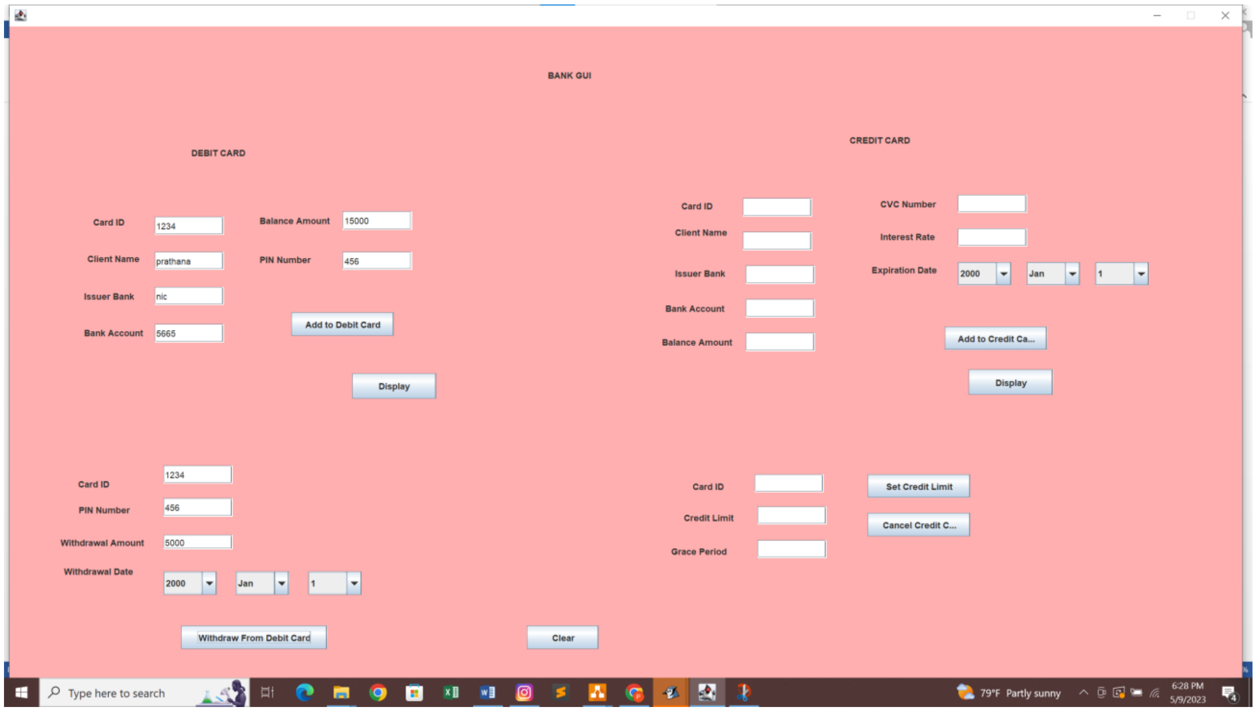
Figure 14 Screenshot of displayed message

Objective :	To input values of balance amount, card Id, bank account, issuer bank, client name, Interest rate in arraylist
Action:	➢ Input values in required field ➢ Press the button
***Expected Result:**	Card has been added
Actual Result:	Inputted values are add to the arraylist
Conclusion:	The test is successful.

Table 12 Test table add to credit card button

WithdrawcashButton

i. Adding values in withdrawal to arraylist

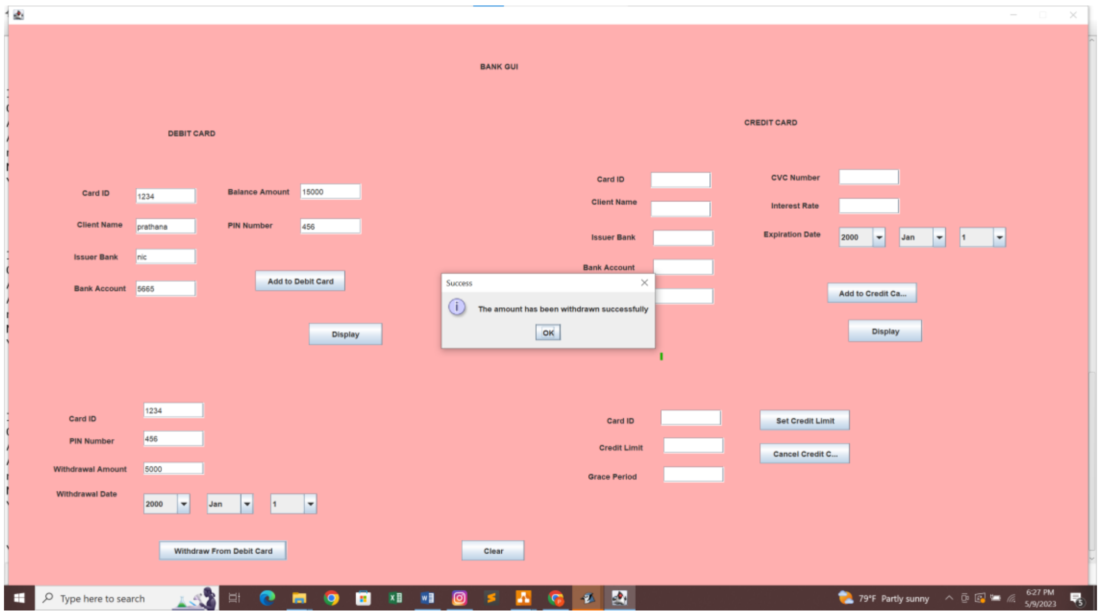


Card ID	1234	Balance Amount	15000
Client Name	prathana	PIN Number	456
Issuer Bank	nic		
Bank Account	5665		

Card ID	1234
PIN Number	456
Withdrawal Amount	5000
Withdrawal Date	2000 Jan 1

Figure 15 Screenshot of withdrawal

ii. Function of withdrawal button after button pressed



Card ID: 1234
Client Name: prathana
Issuer Bank: nic
Bank Account: 5665
Balance Amount: 15000
PIN Number: 456

Card ID:
Client Name:
Issuer Bank:
Bank Account:
CVC Number:
Interest Rate:
Expiration Date: 2000 Jan 1

Card ID: 1234
PIN Number: 456
Withdrawal Amount: 5000
Withdrawal Date: 2000 Jan 1

Card ID:
Credit Limit:
Grace Period:

Success
The amount has been withdrawn successfully
OK

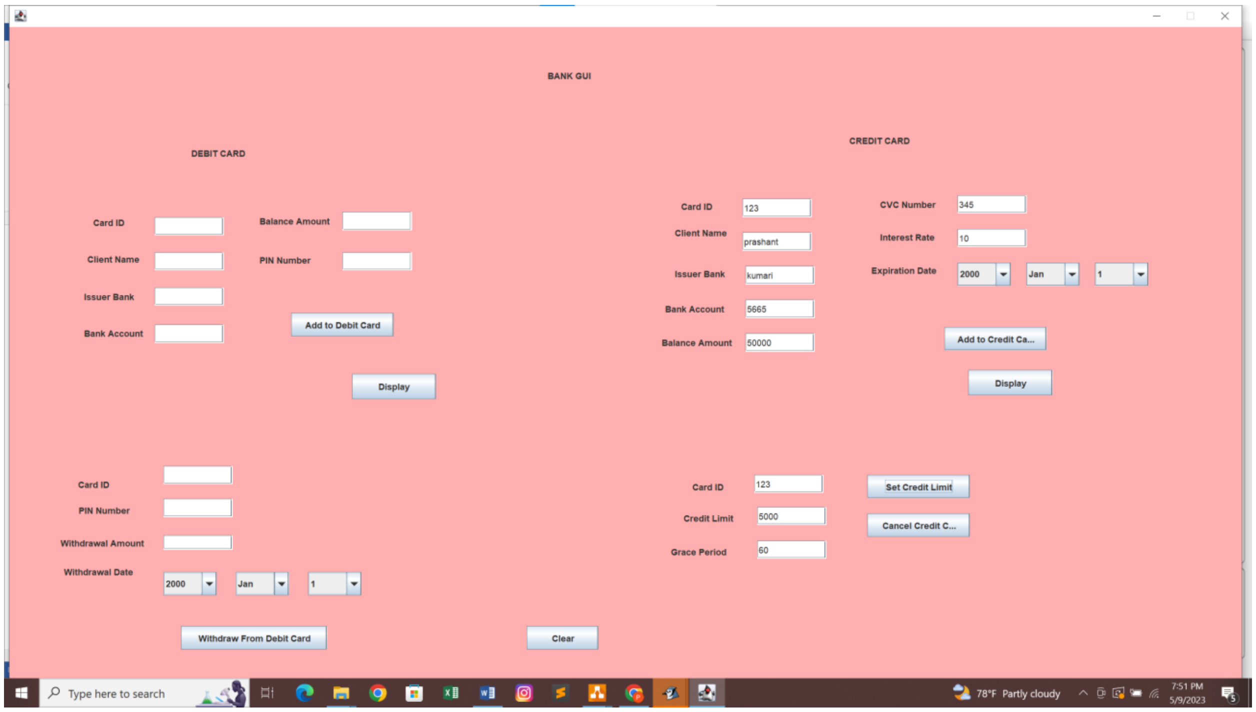
Figure 16 Screenshot of withdrawal after button pressed

Objective :	To input values of balance amount, card Id, bank account, issuer bank, client name, Interest rate in arraylist
Action:	➢ Input values in required field ➢ Press the button
***Expected Result:**	Card has been added
Actual Result:	Inputted values are add to the arraylist
Conclusion:	The test is successful.

Table 13 Test table of withdrawal button

Set credit limit Button

i. Adding values in set credit limit to arraylist



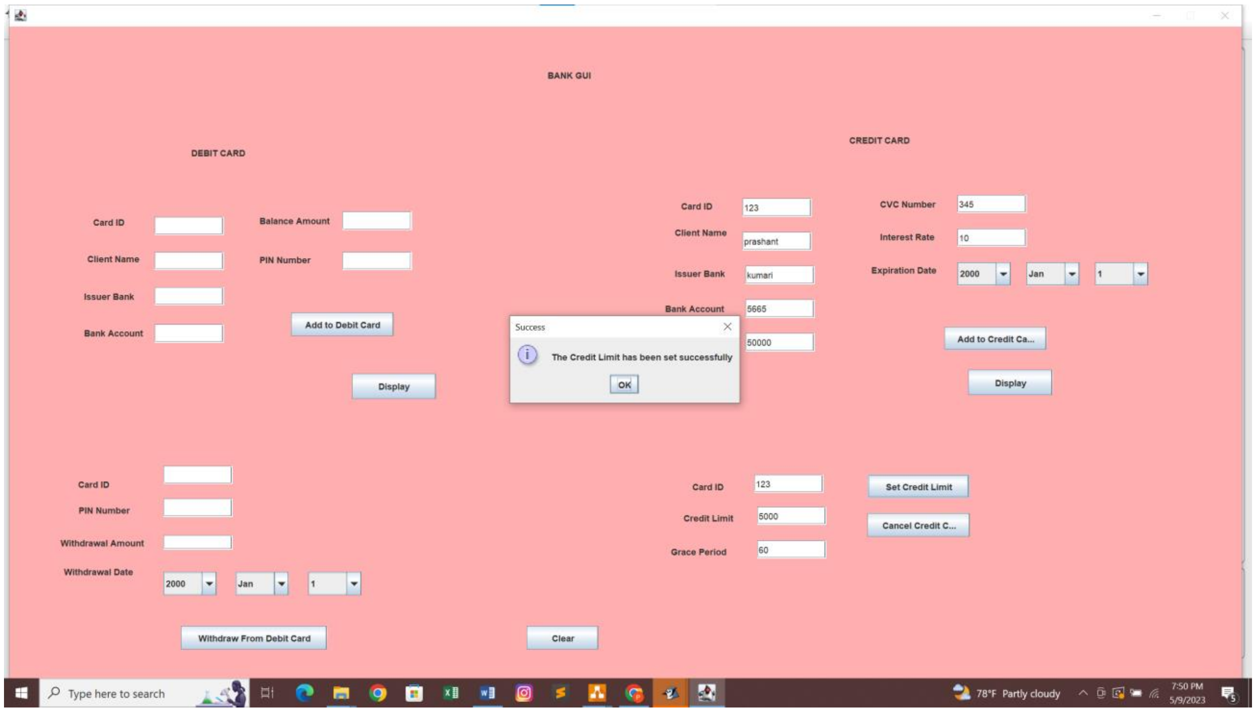
Card ID	123
Client Name	prashant
Issuer Bank	kumari
Bank Account	5665
Balance Amount	50000
CVC Number	345
Interest Rate	10
Expiration Date	2000 Jan 1

Card ID	123
Credit Limit	5000
Grace Period	60

Withdrawal Date	2000 Jan 1

Figure 17 Screenshot of adding in set credit limit button

ii. **Function of set credit limit button after button pressed**



Card ID: 123
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
50000
CVC Number: 345
Interest Rate: 10
Expiration Date: 2000 Jan 1

The Credit Limit has been set successfully

Card ID: 123
Credit Limit: 5000
Grace Period: 60

Withdrawal Date: 2000 Jan 1

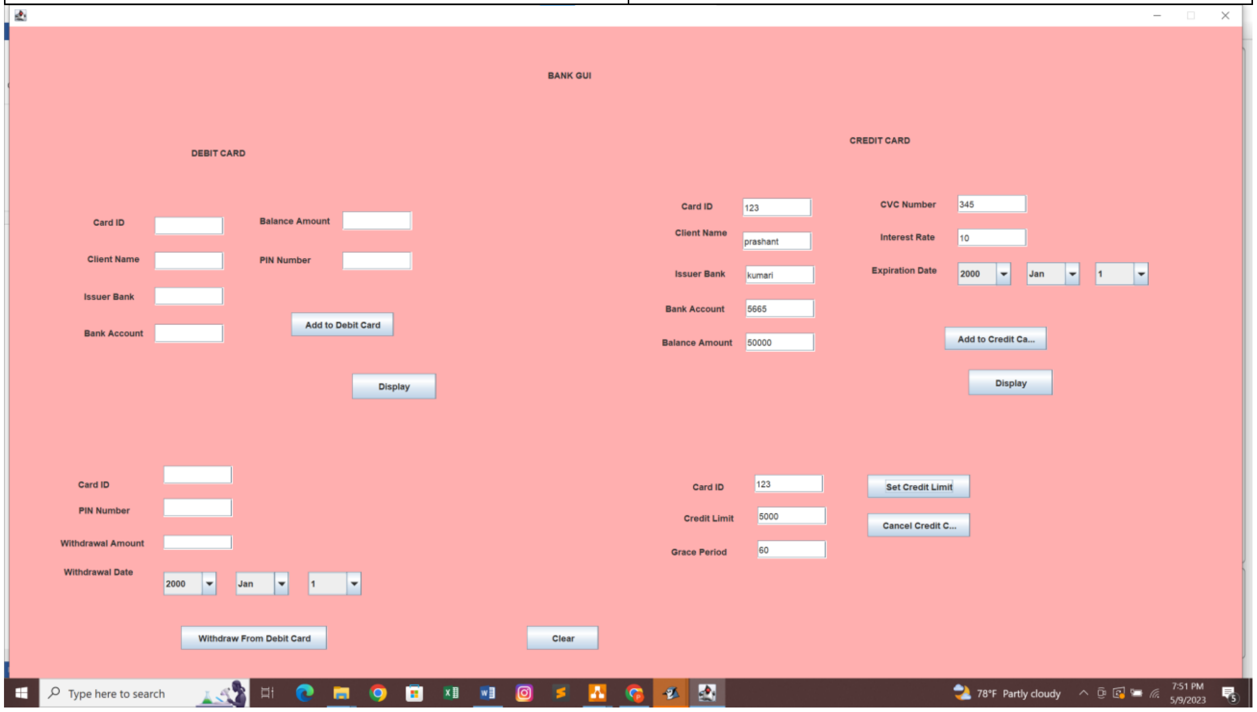
Figure 18 Screenshot of setting credit limit

Table 14 Test table of set credit limit

Cancel credit card Button

i. **Adding values in set credit limit to arraylist**

Objective :	To input values of balance amount, card Id, bank account, issuer bank, client name, Interest rate, Credit limit , grace period in arraylist
Action:	➢ Add to credit card ➢ Input values in required field ➢ Press the button
***Expected Result:**	Credit limit has been set
Actual Result:	Credit limit has been set in arraylist
Conclusion:	The test is successful.



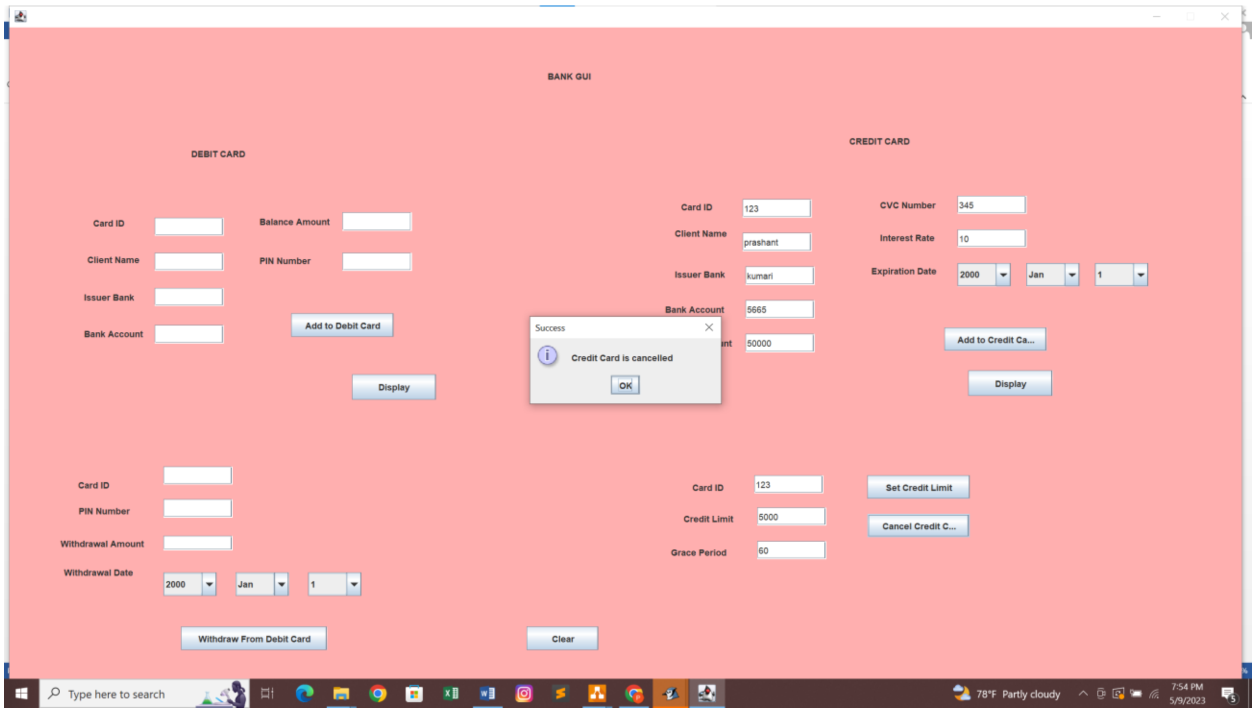
Withdrawal Date: 2000 Jan 1

Card ID: 123
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
Balance Amount: 50000
CVC Number: 345
Interest Rate: 10
Expiration Date: 2000 Jan 1

Card ID: 123
Credit Limit: 5000
Grace Period: 60

Figure 19 Screen shot of adding values in credit card

ii. Function of cancel credit card button after pressed



Card ID: 123
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
50000
CVC Number: 345
Interest Rate: 10
Expiration Date: 2000 Jan 1

Success
Credit Card is cancelled
OK

Card ID: 123
Credit Limit: 5000
Grace Period: 60

Withdrawal Date: 2000 Jan 1

78°F Partly cloudy 7:54 PM 5/9/2023

Figure 20 Screenshot of message after button pressed

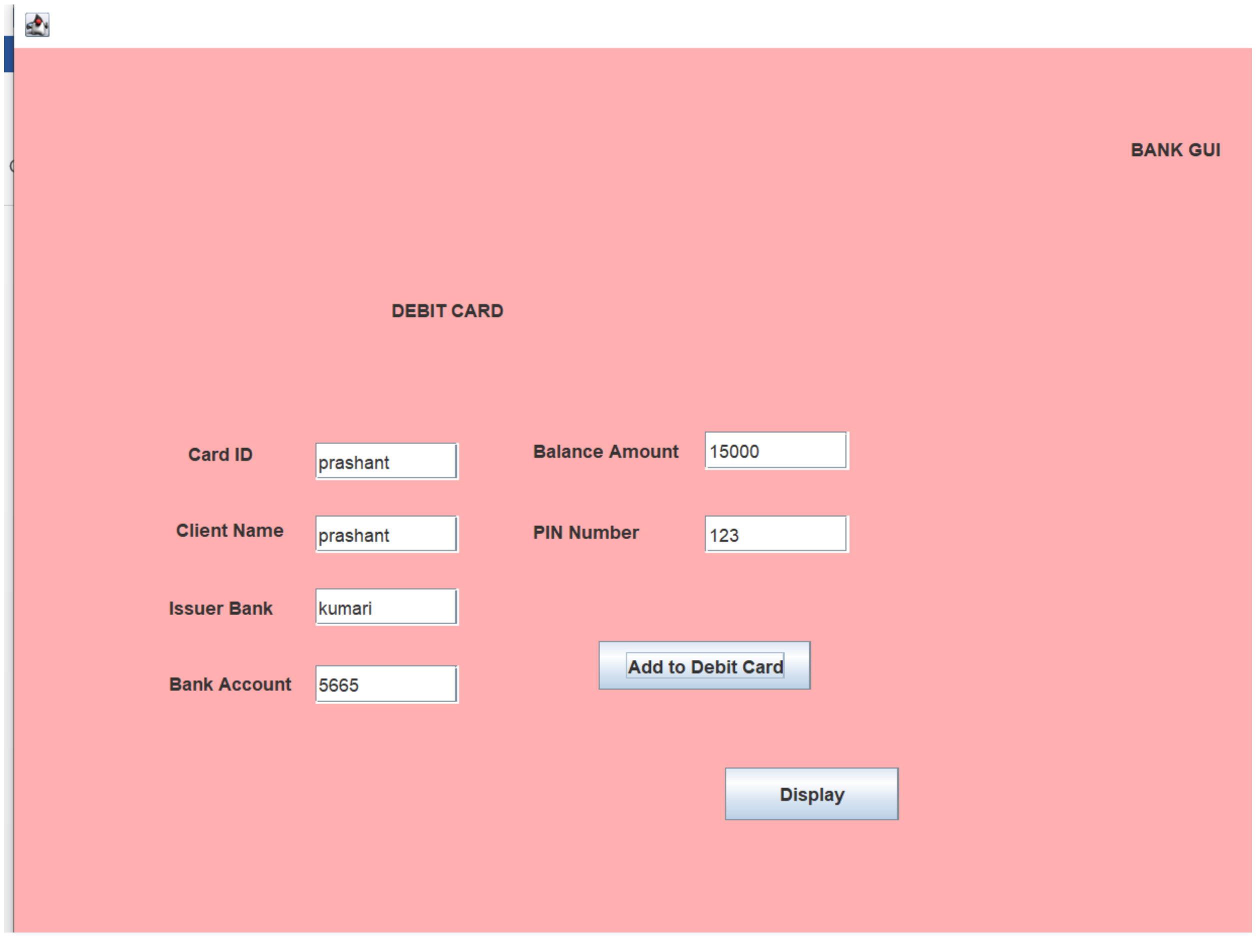
Objective :	To input values of balance amount, card Id, bank account, issuer bank, client name, Interest rate, Credit limit , grace period in arraylist
Action:	➢ Add to credit card ➢ Input values in required field ➢ Press the button
***Expected Result:**	Credit limit has been set
Actual Result:	Credit limit has been set in arraylist

Conclusion:	The test is successful.

Table 15 Table of Cancel credit card button

4.3. Test 3: Test that appropriate dialog boxes appear when unsuitable values are entered for the Card ID:

i. Adding unsuitable value in Card ID of debit card


BANK GUI

DEBIT CARD

Card ID: prashant

Balance Amount: 15000

Client Name: prashant

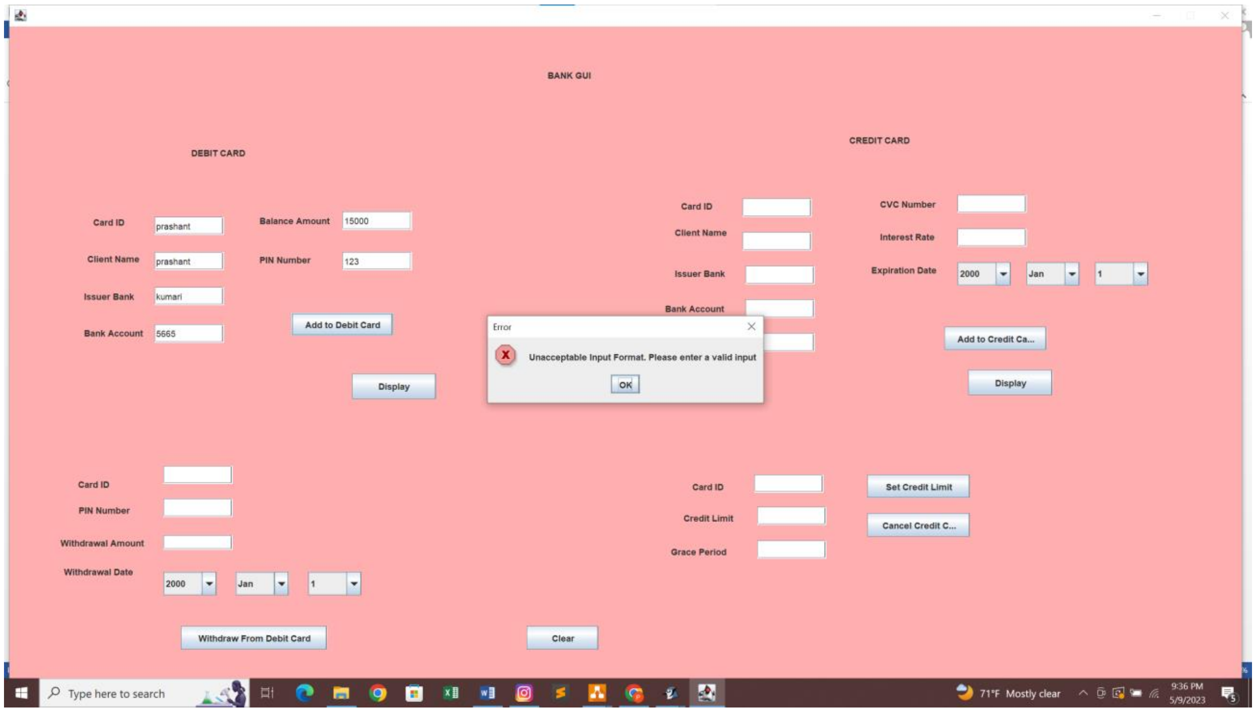
PIN Number: 123

Issuer Bank: kumari

Bank Account: 5665

Figure 21 Screen shot of unsuitable value in debit card

ii. Result after pressing the button with unsuitable value in Card ID



Card ID: prashant
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
Balance Amount: 15000
PIN Number: 123

Expiration Date: 2000 Jan 1
Withdrawal Date: 2000 Jan 1

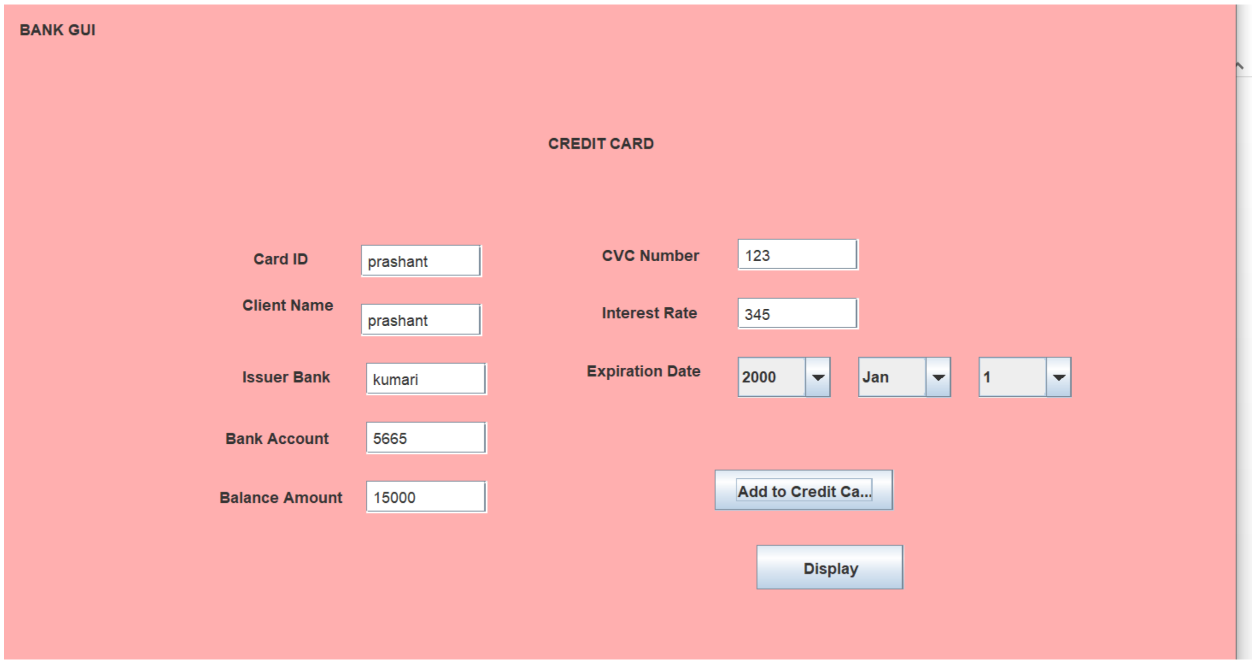
Error
Unacceptable Input Format. Please enter a valid input
OK

Figure 22 Screenshot of result of unsuitable value in Debit card

Objective :	To input unsuitable value (String) in Card ID
Action:	➢ Input unsuitable values (String) in Card ID ➢ Press the button
***Expected Result:**	Unacceptable Input format
Actual Result:	Number format exception
Conclusion:	The test is successful.

Table 16 Test Table unsuitable value in Debit card

i. Adding unsuitable value in Card ID of Credit card

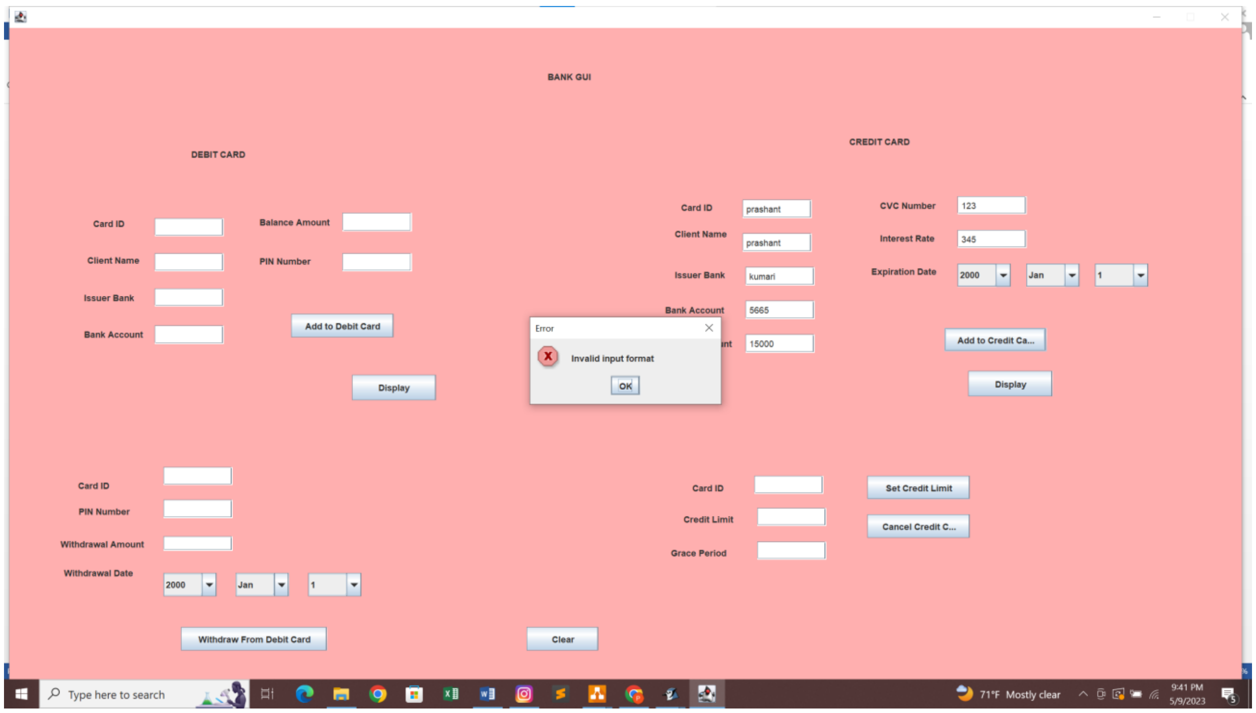

BANK GUI

CREDIT CARD

Card ID	prashant
Client Name	prashant
Issuer Bank	kumari
Bank Account	5665
Balance Amount	15000
CVC Number	123
Interest Rate	345
Expiration Date	2000 Jan 1

Figure 22 Screen shot on unsuitable value in Credit card

ii. Result after pressing the button with unsuitable value in Card ID



Card ID: prashant
Client Name: prashant
Issuer Bank: kumari
Bank Account: 5665
15000
CVC Number: 123
Interest Rate: 345
Expiration Date: 2000 Jan 1

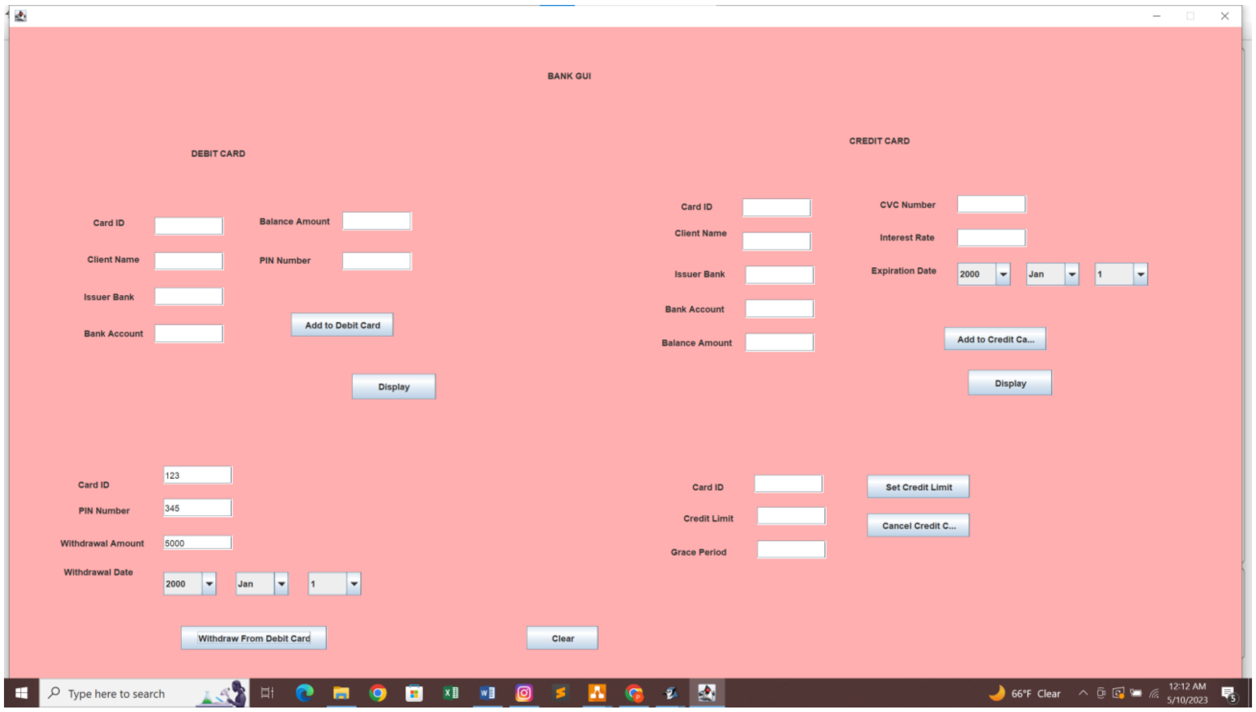
Error
Invalid input format

Figure 23 Screenshot of result of unsuitable value in Credit card

Objective :	To input unsuitable value (String) in Card ID
Action:	➢ Input unsuitable values (String) in Card ID ➢ Press the button
***Expected Result:**	Invalid input format
Actual Result:	Number format exception
Conclusion:	The test is successful.

Table 17 Test Table unsuitable value in Credit card

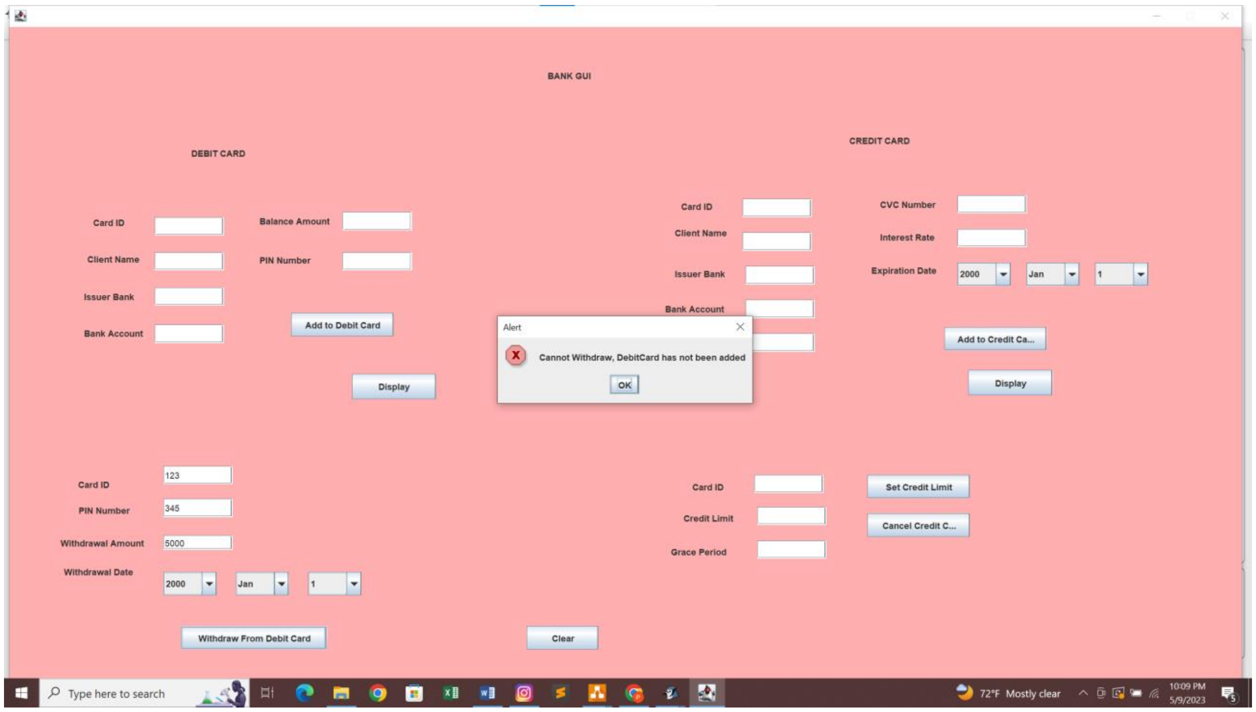
i. Withdrawing money with out adding debit card



Card ID: 123
PIN Number: 345
Withdrawal Amount: 5000
Withdrawal Date: 2000 Jan 1

Figure 23 Withdrawing money without adding debit card

ii. Result after withdrawing money without adding debit card



Alert: Cannot Withdraw, DebitCard has not been added

Card ID: 123
PIN Number: 345
Withdrawal Amount: 5000
Withdrawal Date: 2000 Jan 1

Figure 24 screen shot of result of withdrawing money without adding debit card

Objective :	To input withdraw moneny without adding debit card
Action:	➢ Input Card ID , pin number and withdraw amount ➢ Press the button
***Expected Result:**	Cannot withdraw debit card is not added
Actual Result:	Debit card is not added in arraylist
Conclusion:	The test is successful.

Table 18 table of withdrawing money without adding debit

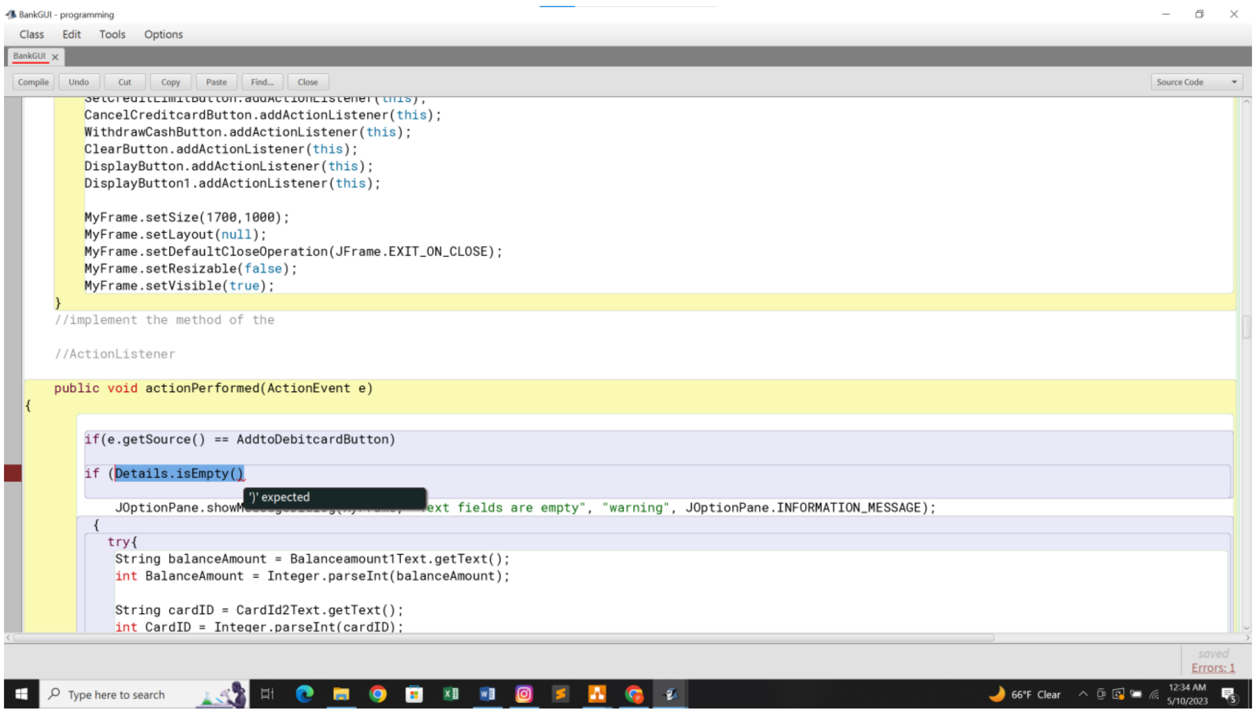
5. Errors

An error in computer data is called Bug. A software bug is an error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

5.1. Syntax Error

Syntax errors are mistakes in the source code, such as spelling and punctuation errors**,** incorrect labels, and so on, which cause an error message to be generated by the compiler.

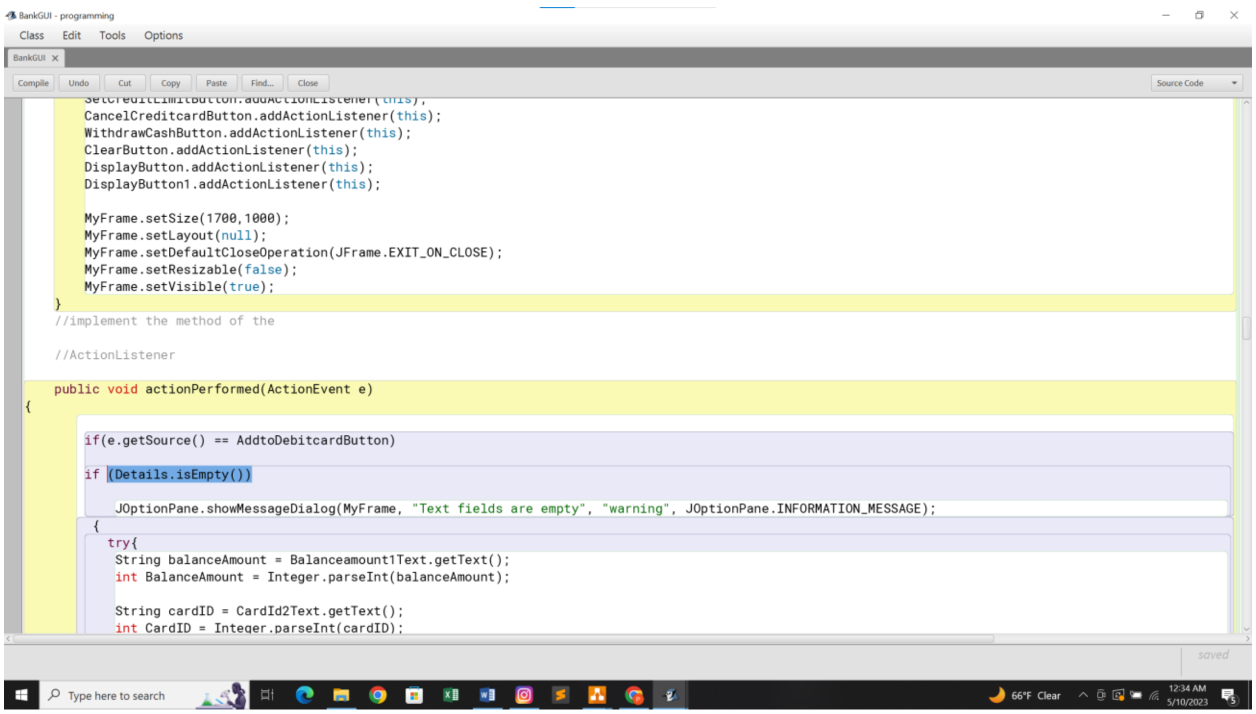
Before rectification



```
        SetCreditLimitButton.addActionListener(this);
        CancelCreditcardButton.addActionListener(this);
        WithdrawCashButton.addActionListener(this);
        ClearButton.addActionListener(this);
        DisplayButton.addActionListener(this);
        DisplayButton1.addActionListener(this);

        MyFrame.setSize(1700,1000);
        MyFrame.setLayout(null);
        MyFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        MyFrame.setResizable(false);
        MyFrame.setVisible(true);
    }
    //implement the method of the

    //ActionListener

    public void actionPerformed(ActionEvent e)
{

        if(e.getSource() == AddtoDebitcardButton)

        if (Details.isEmpty()

            JOptionPane.showM                       ext fields are empty", "warning", JOptionPane.INFORMATION_MESSAGE);
         {
           try{
            String balanceAmount = Balanceamount1Text.getText();
            int BalanceAmount = Integer.parseInt(balanceAmount);

            String cardID = CardId2Text.getText();
            int CardID = Integer.parseInt(cardID);
```

Figure 24 Screenshot of syntax error before rectification

After rectification

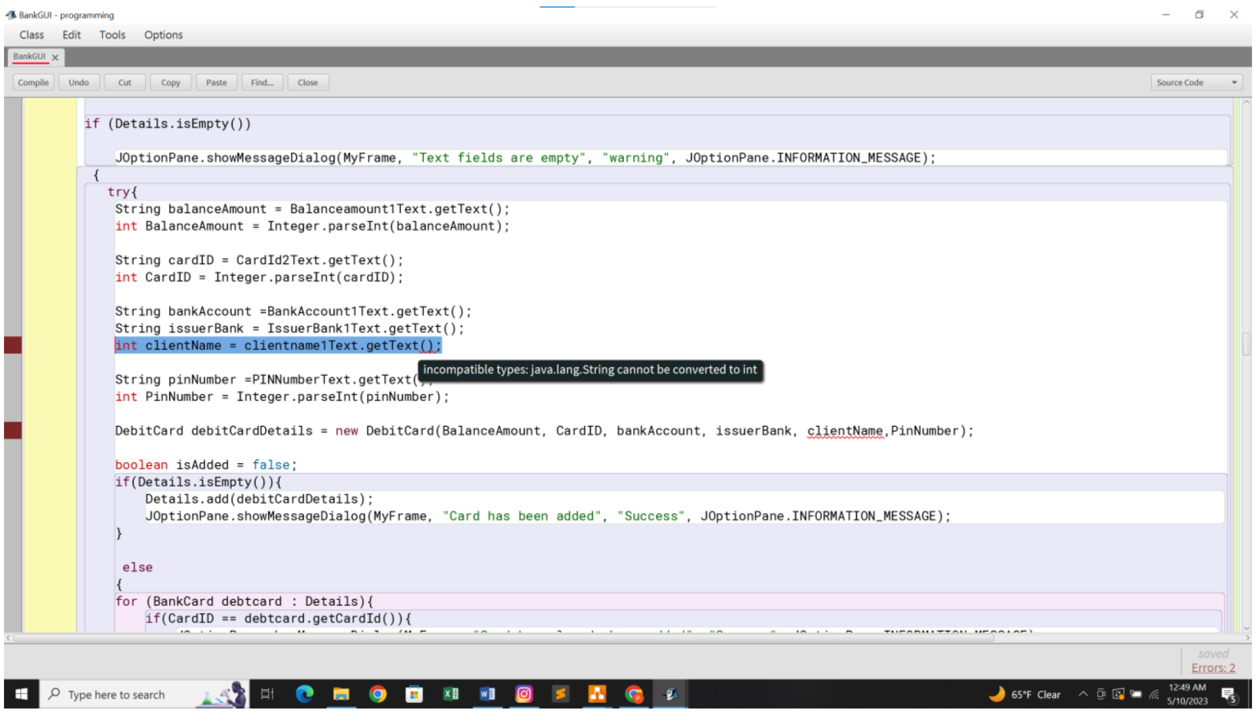


```
        SetCreditLimitButton.addActionListener(this);
        CancelCreditcardButton.addActionListener(this);
        WithdrawCashButton.addActionListener(this);
        ClearButton.addActionListener(this);
        DisplayButton.addActionListener(this);
        DisplayButton1.addActionListener(this);

        MyFrame.setSize(1700,1000);
        MyFrame.setLayout(null);
        MyFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        MyFrame.setResizable(false);
        MyFrame.setVisible(true);
    }
    //implement the method of the

    //ActionListener

    public void actionPerformed(ActionEvent e)
{

        if(e.getSource() == AddtoDebitcardButton)

        if (Details.isEmpty())

            JOptionPane.showMessageDialog(MyFrame, "Text fields are empty", "warning", JOptionPane.INFORMATION_MESSAGE);
          {
            try{
             String balanceAmount = Balanceamount1Text.getText();
             int BalanceAmount = Integer.parseInt(balanceAmount);

             String cardID = CardId2Text.getText();
             int CardID = Integer.parseInt(cardID);
```

Figure 25 Screenshot of syntax error after rectification

5.2. Sematic Error

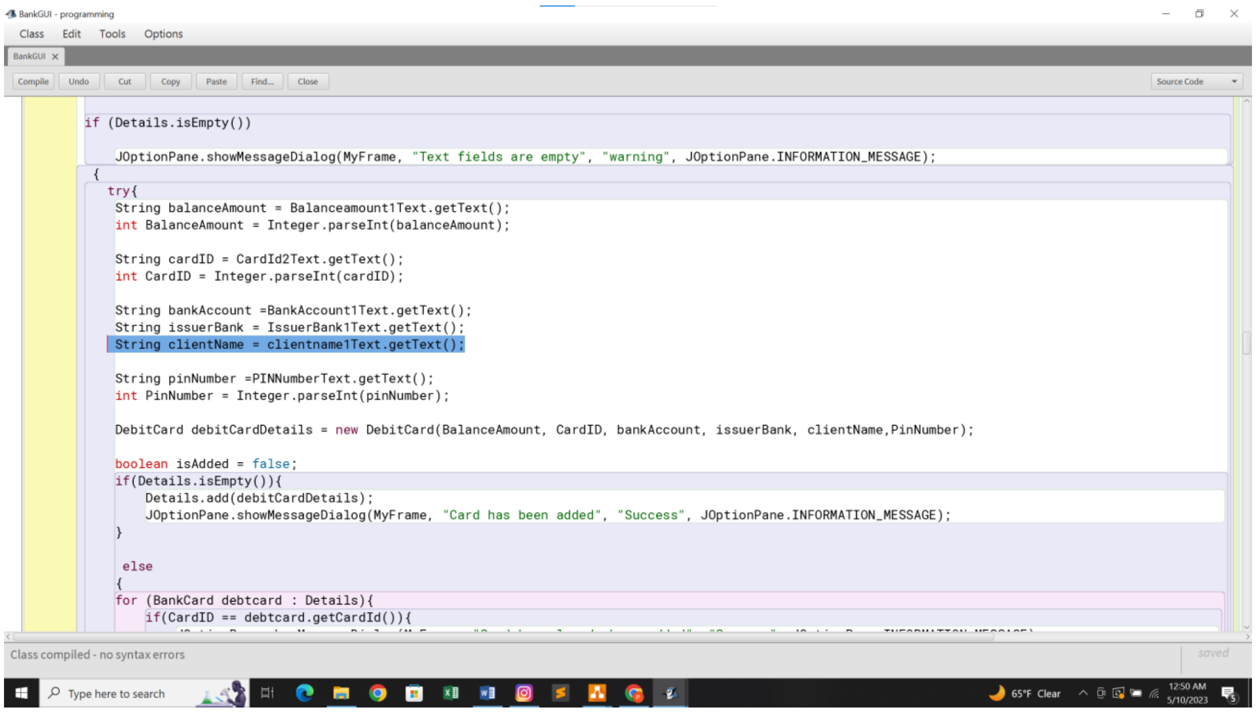
Before rectification



```
if (Details.isEmpty())

    JOptionPane.showMessageDialog(MyFrame, "Text fields are empty", "warning", JOptionPane.INFORMATION_MESSAGE);
 {
   try{
    String balanceAmount = Balanceamount1Text.getText();
    int BalanceAmount = Integer.parseInt(balanceAmount);

    String cardID = CardId2Text.getText();
    int CardID = Integer.parseInt(cardID);

    String bankAccount =BankAccount1Text.getText();
    String issuerBank = IssuerBank1Text.getText();
    int clientName = clientname1Text.getText();
                                                 incompatible types: java.lang.String cannot be converted to int
    String pinNumber =PINNumberText.getText();
    int PinNumber = Integer.parseInt(pinNumber);

    DebitCard debitCardDetails = new DebitCard(BalanceAmount, CardID, bankAccount, issuerBank, clientName,PinNumber);

    boolean isAdded = false;
    if(Details.isEmpty()){
        Details.add(debitCardDetails);
        JOptionPane.showMessageDialog(MyFrame, "Card has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
    }

     else
    {
    for (BankCard debtcard : Details){
        if(CardID == debtcard.getCardId()){
```

Figure 26 Screenshot of sematic error Before rectification

After rectification



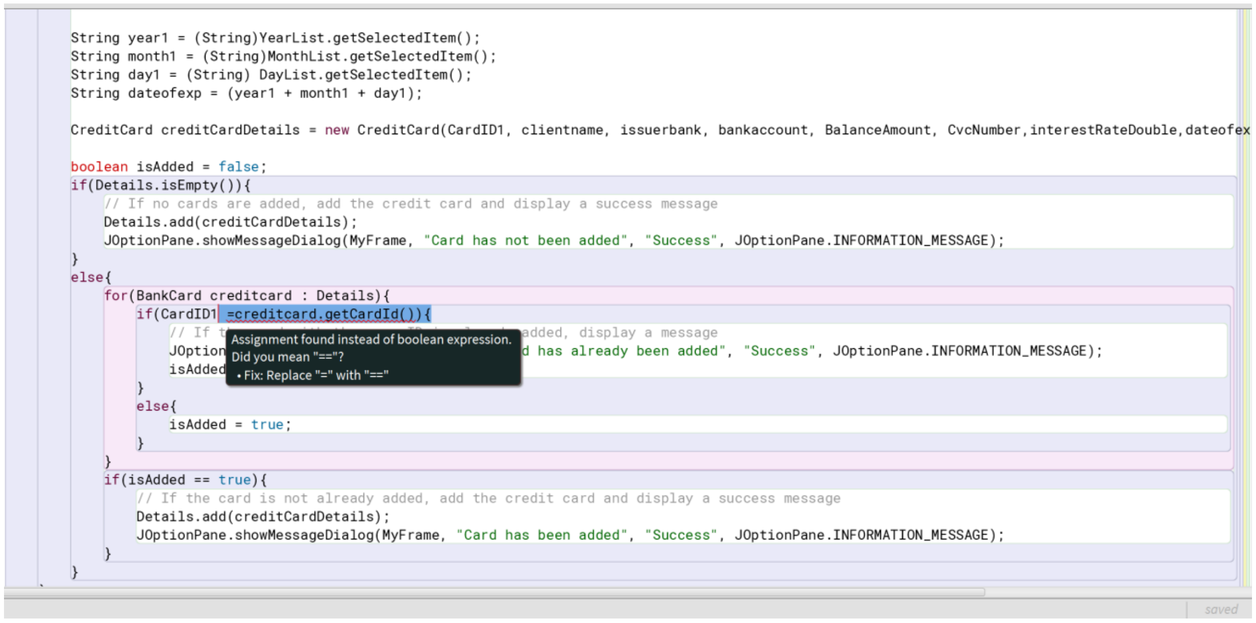
```
if (Details.isEmpty())

    JOptionPane.showMessageDialog(MyFrame, "Text fields are empty", "warning", JOptionPane.INFORMATION_MESSAGE);
 {
   try{
    String balanceAmount = Balanceamount1Text.getText();
    int BalanceAmount = Integer.parseInt(balanceAmount);

    String cardID = CardId2Text.getText();
    int CardID = Integer.parseInt(cardID);

    String bankAccount =BankAccount1Text.getText();
    String issuerBank = IssuerBank1Text.getText();
    String clientName = clientname1Text.getText();

    String pinNumber =PINNumberText.getText();
    int PinNumber = Integer.parseInt(pinNumber);

    DebitCard debitCardDetails = new DebitCard(BalanceAmount, CardID, bankAccount, issuerBank, clientName,PinNumber);

    boolean isAdded = false;
    if(Details.isEmpty()){
        Details.add(debitCardDetails);
        JOptionPane.showMessageDialog(MyFrame, "Card has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
    }

     else
    {
    for (BankCard debtcard : Details){
        if(CardID == debtcard.getCardId()){
```

Class compiled - no syntax errors

Figure 27 Screenshot of sematic error after rectification

5.3. Logical Error

Output for logical error

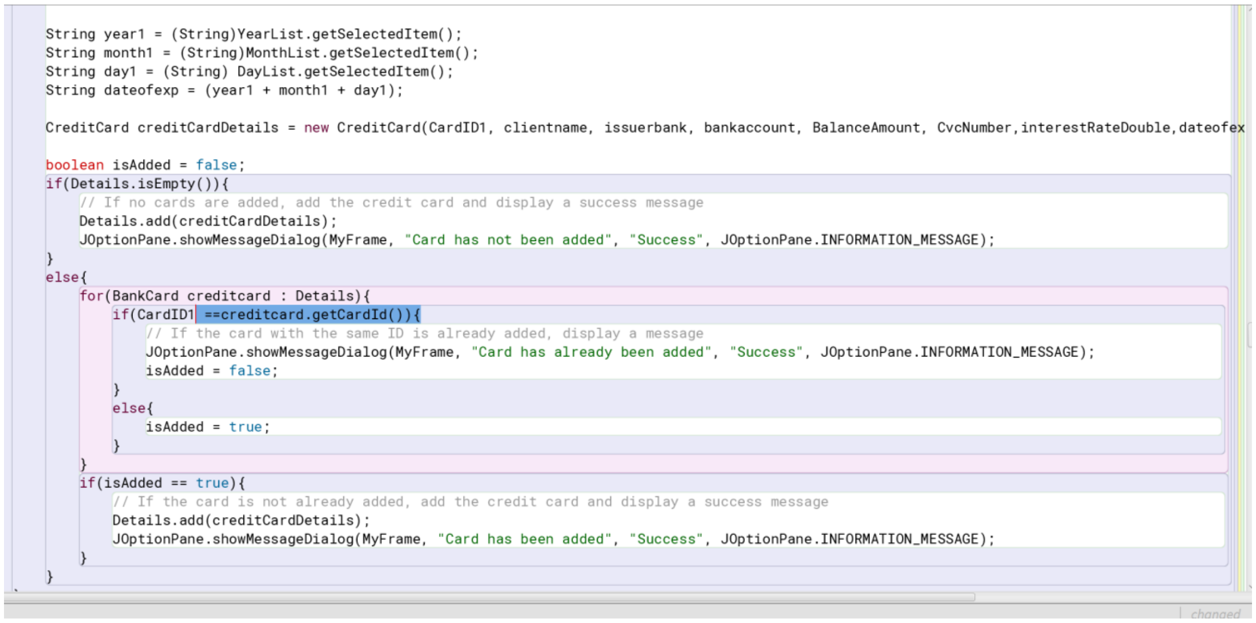


```
String year1 = (String)YearList.getSelectedItem();
String month1 = (String)MonthList.getSelectedItem();
String day1 = (String) DayList.getSelectedItem();
String dateofexp = (year1 + month1 + day1);

CreditCard creditCardDetails = new CreditCard(CardID1, clientname, issuerbank, bankaccount, BalanceAmount, CvcNumber,interestRateDouble,dateofex

boolean isAdded = false;
if(Details.isEmpty()){
    // If no cards are added, add the credit card and display a success message
    Details.add(creditCardDetails);
    JOptionPane.showMessageDialog(MyFrame, "Card has not been added", "Success", JOptionPane.INFORMATION_MESSAGE);
}
else{
    for(BankCard creditcard : Details){
        if(CardID1 =creditcard.getCardId()){
            // If t                              added, display a message
            JOption                              d has already been added", "Success", JOptionPane.INFORMATION_MESSAGE);
            isAdded
        }
        else{
            isAdded = true;
        }
    }
    if(isAdded == true){
        // If the card is not already added, add the credit card and display a success message
        Details.add(creditCardDetails);
        JOptionPane.showMessageDialog(MyFrame, "Card has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
    }
}
```

Assignment found instead of boolean expression.
Did you mean "=="?
• Fix: Replace "=" with "=="

Figure 25 Screenshot of Logical error

Output for correction:



```
String year1 = (String)YearList.getSelectedItem();
String month1 = (String)MonthList.getSelectedItem();
String day1 = (String) DayList.getSelectedItem();
String dateofexp = (year1 + month1 + day1);

CreditCard creditCardDetails = new CreditCard(CardID1, clientname, issuerbank, bankaccount, BalanceAmount, CvcNumber,interestRateDouble,dateofex

boolean isAdded = false;
if(Details.isEmpty()){
    // If no cards are added, add the credit card and display a success message
    Details.add(creditCardDetails);
    JOptionPane.showMessageDialog(MyFrame, "Card has not been added", "Success", JOptionPane.INFORMATION_MESSAGE);
}
else{
    for(BankCard creditcard : Details){
        if(CardID1 ==creditcard.getCardId()){
            // If the card with the same ID is already added, display a message
            JOptionPane.showMessageDialog(MyFrame, "Card has already been added", "Success", JOptionPane.INFORMATION_MESSAGE);
            isAdded = false;
        }
        else{
            isAdded = true;
        }
    }
    if(isAdded == true){
        // If the card is not already added, add the credit card and display a success message
        Details.add(creditCardDetails);
        JOptionPane.showMessageDialog(MyFrame, "Card has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
    }
}
```

Figure 26 Screenshot of logical error

6. Conclusion

The purpose of this assignment was to evaluate the students' ability to use Java programming to create a graphical user interface for a bank card system. The assignment entailed incorporating a class into the project that had already been created in the first section of the coursework and putting in place a user-friendly and aesthetically pleasing GUI that can hold information about bank cards in an arraylist.

This curriculum gave me a lot of insight into real-world challenges. Classes are dependent on one another in a similar way to how people depend on one another in our daily lives. I learned more about catch-block-catchable exceptions like NumberFromat and NullPointer, the GUI, and logical errors. I also realized how difficult it is to fix a logical error. I've also gained a better understanding of inheritance and object-oriented programming. I finally understand how to downcast an item with the brand-new boolean term "instanceof," which provides a result. While working on this project, I encountered a lot of problems. Some of these included simple misinterpretations of the curriculum, while others involved button usability problems.

However the experienced teacher helped me to fix those issues and complete my work on time throughout the course work I gained a lot of programming knowledge. Completing this assignment, I was able to gain valuable experience in software development, debugging, and testing, which are essential skills for a career in related fields.

7. Bibliography

Anon., n.d. [Online]
Available at: https://www.googleadservices.com

Anon., n.d. [Online]
Available at: https://bluej.org/

8. Apendix

8.1. BankGUI

```
import javax.swing.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Color;
public class BankGUI implements ActionListener
{
    //declare all the components here
    private JFrame MyFrame;
    private ArrayList<BankCard> Details = new ArrayList();


    private JButton
AddtoCreditcardButton,SetCreditLimitButton,CancelCreditcardButton,Displ
```

```
ayButton,
DisplayButton1,ClearButton,AddtoDebitcardButton,WithdrawCashButton;


    private JLabel
BankGUILabel,CreditcardLabel,Creditcard,DetailsLabel,CardId1Label,CVC
Number1Label,clientname1Label,IssuerBank1Label,BankAccount1Label,B
alanceamount1Label,

InterestRate1Label,CardId2Label,CreditLimit1Label,graceperiod1Label,Exp
irationDate1Label,DebitcardLabel,CardId3Label,clientname2Label,IssuerB
ank2Label,BankAccount2Label,PINNumber2Label,

Balanceamount2Label,CardId4Label,PINNumber1Label,WithdrawalAmount
Label,WithdrawalDateLabel;


    private JTextField
CardIdText,CVCNumberText,clientnameText,IssuerBankText,BankAccount
Text,BalanceamountText,InterestRateText,CardId1Text,CreditLimitText,

graceperiodText,CardId2Text,clientname1Text,IssuerBank1Text,BankAcco
unt1Text,PINNumberText,Balanceamount1Text,CardId3Text,PINNumber1
Text,WithdrawalAmountText;


    private JComboBox<String>
DayList,MonthList,YearList,Day2List,Month2List,Year2List;


    public BankGUI(){
        //create the code to write GUI
        MyFrame = new JFrame();
        BankGUILabel = new JLabel("BANK GUI");
```

```
BankGUILabel.setBounds(736,36,97,60);
MyFrame.getContentPane().setBackground(Color.PINK);

// ----creating Credit Card components

//JLabel for Credit Card
CreditcardLabel = new JLabel("CREDIT CARD ");
CardId1Label = new JLabel("Card ID");
CVCNumber1Label = new JLabel("CVC Number");
clientname1Label = new JLabel("Client Name");
IssuerBank1Label = new JLabel("Issuer Bank");
BankAccount1Label = new JLabel("Bank Account");
Balanceamount1Label = new JLabel("Balance Amount");
InterestRate1Label = new JLabel("Interest Rate");
CardId2Label = new JLabel("Card ID");
CreditLimit1Label = new JLabel("Credit Limit");
graceperiod1Label = new JLabel("Grace Period");
ExpirationDate1Label = new JLabel("Expiration Date");

//setBounds for Credit Card jlabel
CreditcardLabel.setBounds(1149,131,117,48);
CardId1Label.setBounds(919,228,75,35);
CardId2Label.setBounds(934,616,55,25);
CVCNumber1Label.setBounds(1191,225,88,35);
clientname1Label.setBounds(910,271,93,20);
```

```
IssuerBank1Label.setBounds(910,327,97,20);
BankAccount1Label.setBounds(897,368,97,35);
Balanceamount1Label.setBounds(892,414,102,35);
InterestRate1Label.setBounds(1191,270,93,35);
ExpirationDate1Label.setBounds(1179,315,105,35);
CreditLimit1Label.setBounds(922,661,79,20);
graceperiod1Label.setBounds(905,707,99,20);

//JTextField for Credit Card
CardIdText = new JTextField();
CVCNumberText = new JTextField();
clientnameText = new JTextField();
IssuerBankText = new JTextField();
BankAccountText = new JTextField();
BalanceamountText = new JTextField();
InterestRateText = new JTextField();
CardId1Text = new JTextField();
CreditLimitText = new JTextField();
graceperiodText = new JTextField();

//setBounds for Credit Card jtextfield
CardIdText.setBounds(1003,235,95,25);
clientnameText.setBounds(1003,281,95,25);
IssuerBankText.setBounds(1007,327,95,25);
BankAccountText.setBounds(1007,373,95,25);
```

```
        BalanceamountText.setBounds(1007,419,95,25);
        CVCNumberText.setBounds(1297,230,95,25);
        InterestRateText.setBounds(1297,276,95,25);
        CardId1Text.setBounds(1019,612,95,25);
        CreditLimitText.setBounds(1023,656,95,25);
        graceperiodText.setBounds(1023,702,95,25);

        //JComboBox for Credit Card
        String[] Years =
{"2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2
010","2011","2012","2013","2014","2015","2016","2017","2018","2019","202
0","2021","2022","2023"};
        YearList = new JComboBox<String>(Years);
        String[] Months =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sept","Oct","Nov","Dec"};
        MonthList = new JComboBox<String>(Months);
        String[] Days =
{"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18",
"19","20","21","22","23","24","25","26","27","28","29","30","31"};
        DayList = new JComboBox<String>(Days);

        //setBounds for Credit Card jcombo box
        YearList.setBounds(1297,322,73,32);
        MonthList.setBounds(1391,322,73,32);
        DayList.setBounds(1485,322,73,32);

        //adding the Label of Credit Card
```

```
MyFrame.add(CreditcardLabel);
MyFrame.add(CardId1Label);
MyFrame.add(CardId2Label);
MyFrame.add(CVCNumber1Label);
MyFrame.add(clientname1Label);
MyFrame.add(ExpirationDate1Label);
MyFrame.add(IssuerBank1Label);
MyFrame.add(BankAccount1Label);
MyFrame.add(Balanceamount1Label);;
MyFrame.add(InterestRate1Label);
MyFrame.add(ExpirationDate1Label);
MyFrame.add(CreditLimit1Label);
MyFrame.add(graceperiod1Label);
MyFrame.add(BankGUILabel);
MyFrame.add(CreditcardLabel);

//adding the TextField of Credit Card
MyFrame.add(CardIdText);
MyFrame.add(CardId1Text);
MyFrame.add(CVCNumberText);
MyFrame.add(clientnameText);
MyFrame.add(IssuerBankText);
MyFrame.add(BankAccountText);
MyFrame.add(BalanceamountText);
MyFrame.add(InterestRateText);
MyFrame.add(CreditLimitText);
MyFrame.add(graceperiodText);

// adding credit card combo box
MyFrame.add(DayList);
MyFrame.add(MonthList);
MyFrame.add(YearList);

// ----creating debit card components
//JLabel for Debit Card
DebitcardLabel = new JLabel("DEBIT CARD ");
CardId3Label = new JLabel("Card ID");
clientname1Label = new JLabel("Client Name");
IssuerBank1Label = new JLabel("Issuer Bank");
BankAccount1Label = new JLabel("Bank Account");
PINNumber2Label = new JLabel("PIN Number");
Balanceamount1Label = new JLabel("Balance Amount");
CardId4Label = new JLabel("Card ID");
PINNumber1Label = new JLabel("PIN Number");
WithdrawalAmountLabel = new JLabel("Withdrawal Amount");
WithdrawalDateLabel = new JLabel("Withdrawal Date");

//setBounds for Debit Card jlabel


DebitcardLabel.setBounds(249,155,114,35);
```

```
CardId3Label.setBounds(115,250,75,35);
CardId4Label.setBounds(94,615,44,20);
clientname1Label.setBounds(107,300,93,35);
IssuerBank1Label.setBounds(102,358,97,20);
BankAccount1Label.setBounds(102,401,97,35);
Balanceamount1Label.setBounds(342,248,115,35);
PINNumber2Label.setBounds(342,308,95,20);
PINNumber1Label.setBounds(95,650,95,20);
WithdrawalAmountLabel.setBounds(70,695,124,20);
WithdrawalDateLabel.setBounds(75,731,115,26);

//JTextField for Debit Card
CardId2Text = new JTextField();
clientname1Text = new JTextField();
IssuerBank1Text = new JTextField();
BankAccount1Text = new JTextField();
PINNumberText = new JTextField();
Balanceamount1Text = new JTextField();
CardId3Text = new JTextField();
PINNumber1Text = new JTextField();
WithdrawalAmountText = new JTextField();

//set bounds for debit card jtextfield
CardId2Text.setBounds(199,260,95,25);
clientname1Text.setBounds(199,308,95,25);
```

```
        IssuerBank1Text.setBounds(199,356,95,25);
        BankAccount1Text.setBounds(199,407,95,25);
        PINNumberText.setBounds(456,308,95,25);
        Balanceamount1Text.setBounds(456,253,95,25);
        CardId3Text.setBounds(211,600,95,25);
        PINNumber1Text.setBounds(211,645,95,25);
        WithdrawalAmountText.setBounds(211,695,95,20);

        //JComboBox for Debit Card
        String[] Year = {
"2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2
010","2011","2012","2013","2014","2015","2016","2017","2018","2019","202
0","2021","2022","2023"};
        Year2List = new JComboBox<String>(Year);
        String[] Month = {
"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sept","Oct","Nov","Dec"};
        Month2List = new JComboBox<String>(Month);
        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","31"};
        Day2List = new JComboBox<String>(Day);

        //set bounds for debitcard combo box
        Day2List.setBounds(409,745,73,32);
        Month2List.setBounds(310,745,73,32);
        Year2List.setBounds(211,745,73,32);
```

```
//adding the Label of Debit Card
MyFrame.add(DebitcardLabel);
MyFrame.add(CardId3Label);
MyFrame.add(CardId4Label);
MyFrame.add(clientname1Label);
MyFrame.add(IssuerBank1Label);
MyFrame.add(BankAccount1Label);
MyFrame.add(Balanceamount1Label);
MyFrame.add(PINNumber1Label);
MyFrame.add(PINNumber2Label);
MyFrame.add(WithdrawalAmountLabel);
MyFrame.add(WithdrawalDateLabel);

// adding text field of debitcard
MyFrame.add(CardId2Text);
MyFrame.add(CardId3Text);
MyFrame.add(clientname1Text);
MyFrame.add(IssuerBank1Text);
MyFrame.add(BankAccount1Text);
MyFrame.add(Balanceamount1Text);
MyFrame.add(PINNumberText);
MyFrame.add(PINNumber1Text);
MyFrame.add(WithdrawalAmountText);

//adding the JComboBox for debit card
```

```
MyFrame.add(Day2List);
MyFrame.add(Month2List);
MyFrame.add(Year2List);

//JButton for Credit Card and Debit card
AddtoCreditcardButton = new JButton("Add to Credit Card");
SetCreditLimitButton = new JButton("Set Credit Limit");
CancelCreditcardButton = new JButton("Cancel Credit Card");
DisplayButton = new JButton("Display");
DisplayButton1 = new JButton("Display");
ClearButton = new JButton("Clear");
AddtoDebitcardButton = new JButton("Add to Debit Card");
WithdrawCashButton = new JButton("Withdraw From Debit Card");

//setBounds for jbutton
AddtoDebitcardButton.setBounds(386,391,140,32);
WithdrawCashButton.setBounds(235,819,200,32);
AddtoCreditcardButton.setBounds(1279,410,140,32);
SetCreditLimitButton.setBounds(1174,612,140,32);
CancelCreditcardButton.setBounds(1174,665,140,32);
DisplayButton.setBounds(469,474,115,35);
DisplayButton1.setBounds(1312,469,115,35);
ClearButton.setBounds(708,819,98,32);

//adding the JButton to the JFrame for Credit Card and Debit Card
```

```
        MyFrame.add(AddtoCreditcardButton);
        MyFrame.add(AddtoDebitcardButton);
        MyFrame.add(SetCreditLimitButton);
        MyFrame.add(CancelCreditcardButton);
        MyFrame.add(WithdrawCashButton);
        MyFrame.add(ClearButton);
        MyFrame.add(DisplayButton);
        MyFrame.add(DisplayButton1);

        //
        AddtoDebitcardButton.addActionListener(this);
        AddtoCreditcardButton.addActionListener(this);
        SetCreditLimitButton.addActionListener(this);
        CancelCreditcardButton.addActionListener(this);
        WithdrawCashButton.addActionListener(this);
        ClearButton.addActionListener(this);
        DisplayButton.addActionListener(this);
        DisplayButton1.addActionListener(this);

        MyFrame.setSize(1700,1000);
        MyFrame.setLayout(null);
        MyFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        MyFrame.setResizable(false);
        MyFrame.setVisible(true);
    }
    //implement the method of the


    //ActionListener


    public void actionPerformed(ActionEvent e)
{
         // Check if the source of the event is the "AddtoDebitcardButton"
button
        if(e.getSource() == AddtoDebitcardButton)
          // Check if the "Details" list is empty
        if (Details.isEmpty())
               // Show a message dialog if the text fields are empty
            JOptionPane.showMessageDialog(MyFrame, "Text fields are
empty", "warning", JOptionPane.INFORMATION_MESSAGE);
         {
           try{
               // Extract information from the text fields and convert them to their
corresponding data types
            String balanceAmount = Balanceamount1Text.getText();
            int BalanceAmount = Integer.parseInt(balanceAmount);


            String cardID = CardId2Text.getText();
            int CardID = Integer.parseInt(cardID);


            String bankAccount =BankAccount1Text.getText();
            String issuerBank = IssuerBank1Text.getText();
            String clientName = clientname1Text.getText();


            String pinNumber =PINNumberText.getText();

            int PinNumber = Integer.parseInt(pinNumber);

             // Create a new DebitCard object using the extracted information

            DebitCard debitCardDetails = new DebitCard(BalanceAmount,
CardID, bankAccount, issuerBank, clientName,PinNumber);


            boolean isAdded = false;

             // Create a new DebitCard object using the extracted information

            if(Details.isEmpty()){

                Details.add(debitCardDetails);

                JOptionPane.showMessageDialog(MyFrame, "Card has been
added", "Success", JOptionPane.INFORMATION_MESSAGE);

            }


             else

            {

              // If the "Details" list is not empty, loop through each element of the
list and check if the current card ID matches any previously added card ID

            for (BankCard debtcard : Details){

                if(CardID == debtcard.getCardId()){

                     // If there is a match, show a message dialog indicating that
the card has already been added

                    JOptionPane.showMessageDialog(MyFrame, "Card has
already been added", "Success",
JOptionPane.INFORMATION_MESSAGE);
```

```
                isAdded = false;
            }
            else{
                isAdded = true;
            }
        }
         // If there is no match, add the current DebitCard object to the
"Details" list and show a success message dialog
        if(isAdded == true){
            Details.add(debitCardDetails);
            JOptionPane.showMessageDialog(MyFrame, "Card is not
added", "Success", JOptionPane.INFORMATION_MESSAGE);
        }
         }
        }
      catch(NumberFormatException nfe){
            // Catch a NumberFormatException and show an error message
dialog if the user enters an invalid input format
    JOptionPane.showMessageDialog(MyFrame, "Unacceptable Input
Format. Please enter a valid input", "Error",
JOptionPane.ERROR_MESSAGE);
    }
   }
    //Function to Withdraw Cash Button



    if(e.getSource() == AddtoCreditcardButton){
        try{
            //Get values from text fields
            String cardid = CardIdText.getText();
            int CardID1 = Integer.parseInt(cardid);

            String clientname = clientnameText.getText();
            String issuerbank = IssuerBankText.getText();
            String bankaccount = BankAccountText.getText();

            String balanceamount = BalanceamountText.getText();
            int BalanceAmount = Integer.parseInt(balanceamount);

            String cvcNumber = CVCNumberText.getText();
            int CvcNumber = Integer.parseInt(cvcNumber);

            String interestRate = InterestRateText.getText();
            double interestRateDouble = Double.parseDouble(interestRate);

            String year1 = (String)YearList.getSelectedItem();
            String month1 = (String)MonthList.getSelectedItem();
            String day1 = (String) DayList.getSelectedItem();
            String dateofexp = (year1 + month1 + day1);

            CreditCard creditCardDetails = new CreditCard(CardID1,
clientname, issuerbank, bankaccount, BalanceAmount,
CvcNumber,interestRateDouble,dateofexp);
```

```
        boolean isAdded = false;
        if(Details.isEmpty()){
            // If no cards are added, add the credit card and display a
success message
            Details.add(creditCardDetails);
            JOptionPane.showMessageDialog(MyFrame, "Card has not
been added", "Success", JOptionPane.INFORMATION_MESSAGE);
        }
        else{
            for(BankCard creditcard : Details){
                if(CardID1 ==creditcard.getCardId()){
                    // If the card with the same ID is already added, display a
message
                    JOptionPane.showMessageDialog(MyFrame, "Card has
already been added", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    isAdded = false;
                }
                else{
                    isAdded = true;
                }
            }
            if(isAdded == true){
                // If the card is not already added, add the credit card and
display a success message
                Details.add(creditCardDetails);
```

```
                    JOptionPane.showMessageDialog(MyFrame, "Card has
been added", "Success", JOptionPane.INFORMATION_MESSAGE);
                }
            }
        }
        catch(NumberFormatException nfe){
            JOptionPane.showMessageDialog(MyFrame, "Invalid input
format", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }


    //function of Display button
    // This code block is executed when DisplayButton1 is clicked
     if(e.getSource() == DisplayButton1){
         // Check if there are any CreditCards added
        if(Details.isEmpty()){
              // Display an error message if there are no CreditCards
            JOptionPane.showMessageDialog(MyFrame, "Sorry, no
CreditCard has been added ", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else{
             // Iterate through the list of BankCards to display CreditCard
details
            for(BankCard displaycredit: Details){
                // Check if the current card is a CreditCard
                if(displaycredit instanceof CreditCard){
                      // Display a success message before showing the details
                    JOptionPane.showMessageDialog(MyFrame, "The details of
Credit Card has been displayed", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    System.out.println("\n");
                    // Call the display method for the CreditCard object
                    ((CreditCard)displaycredit).display();
                    System.out.println("\n");
                }
            }
        }

    }

    //function of setcreditlimit button

    // This code block is executed when SetCreditLimitButton is clicked
    if(e.getSource() == SetCreditLimitButton){
        try{
            //Get values from text fields
            String cardid1 = CardId1Text.getText();
            int Cardid1 = Integer.parseInt(cardid1);

            String CreditLimit = CreditLimitText.getText();
            int creditLimit = Integer.parseInt(CreditLimit);

            String gracePeriod = graceperiodText.getText();
            int GracePeriod = Integer.parseInt(gracePeriod);
            // Check if there are any CreditCards added
            if(Details.isEmpty()){
                // Display an error message if there are no CreditCards
               JOptionPane.showMessageDialog(MyFrame, "Cannot set 
Credit Limit ","Alert", JOptionPane.ERROR_MESSAGE);
            }
            else{
                 // Iterate through the list of BankCards to find the CreditCard 
with the specified ID
               for(BankCard setCredit: Details){
                  if(setCredit instanceof CreditCard){
                     if(Cardid1 == setCredit.getCardId()){
                          // Set the Credit Limit and Grace Period for the 
CreditCard
                        ((CreditCard)setCredit).setCreditLimit(creditLimit, 
GracePeriod);
                        JOptionPane.showMessageDialog(MyFrame, "The 
Credit Limit has been set successfully", "Success", 
JOptionPane.INFORMATION_MESSAGE);
                     }
                     else{
                        // Display an error message if the provided ID is not 
found
                        JOptionPane.showMessageDialog(MyFrame, "The 
provided ID has not been found", "Alert", 
JOptionPane.ERROR_MESSAGE);
                     }
```

```
                    }
                    else{
                         // Display an error message if there are no CreditCards
                       JOptionPane.showMessageDialog(MyFrame, "Credit
Card Not Found", "Alert", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        }
        catch(NumberFormatException nfe){
            // Display an error message if the input values are not valid

            JOptionPane.showMessageDialog(MyFrame, "The information
you provided cannot be accepted", "Alert",
JOptionPane.ERROR_MESSAGE);
        }
    }

    // function of cancel credit card button
    // This code block is executed when CancelCreditcardButton is clicked
    if(e.getSource() == CancelCreditcardButton){
        try{
            //Get values from text fields
            String cardid2 = CardId1Text.getText();
            int Cardid1 = Integer.parseInt(cardid2);
        // Check if there are any CreditCards added


        if(Details.isEmpty()){
            // Display an error message if there are no CreditCards
          JOptionPane.showMessageDialog(MyFrame, "Cannot set
Credit Limit, Credit Card is not added", "Alert",
JOptionPane.ERROR_MESSAGE);
        }
        else{
            // Iterate through the list of BankCards to find the
CreditCard with the specified ID
          for(BankCard cancelcard: Details){
            if(cancelcard instanceof CreditCard){
              if(Cardid1 == cancelcard.getCardId()){
                // Cancel the CreditCard
                ((CreditCard)cancelcard).cancelCreditCard();
                JOptionPane.showMessageDialog(MyFrame, "Credit
Card is cancelled", "Success", JOptionPane.INFORMATION_MESSAGE);
              }
              else{
                  // Display an error message if the provided ID is not
found
                JOptionPane.showMessageDialog(MyFrame, "card Id
provided does not exist.", "Success",
JOptionPane.INFORMATION_MESSAGE);
              }
            }
            else{
```

```
                        // Display an error message if there are no CreditCards
                     JOptionPane.showMessageDialog(MyFrame, "Credit
Card Not Found", "Alert", JOptionPane.INFORMATION_MESSAGE);
                  }
               }
            }
         }
         catch(NumberFormatException nfe){
               // Display an error message if the input values are not valid
            JOptionPane.showMessageDialog(MyFrame, "information
cannot be accepted", "Alert", JOptionPane.ERROR_MESSAGE);
         }
      }


      if(e.getSource() == SetCreditLimitButton)
      {
         if(CardId1Text.getText().isEmpty() ||
CreditLimitText.getText().isEmpty() || graceperiodText.getText().isEmpty())
         {
            JOptionPane.showMessageDialog(MyFrame,"Text
Empty","Alert",JOptionPane.ERROR_MESSAGE);
         }
```

```
        else
        {
           try{
              int cardid = Integer.parseInt(CardId1Text.getText());
              double creditlimit =
Double.parseDouble(CreditLimitText.getText());
              int newgraceperiod =
Integer.parseInt(graceperiodText.getText());
              boolean car = false;
              for(BankCard credit : Details)
              {
                 if(credit instanceof CreditCard)
                 {
                    CreditCard c =(CreditCard) credit;
                    if(cardid == c.getCardId())
                    {
                       car = true;
                       if(creditlimit <= 2.5* c.getBalanceAmount())
                       {
                          c.setCreditLimit(cardid,newgraceperiod);

JOptionPane.showMessageDialog(MyFrame,"Successfull","Alert",JOptionP
ane.INFORMATION_MESSAGE);
                       }
                       else{
                          JOptionPane.showMessageDialog(MyFrame,"Credit
Limit is too high","Alert",JOptionPane.ERROR_MESSAGE);
```

```
                        }
                        break;
                    }
                    else
                    {
                        car = false;
                    }
                }
            }
            if (car==false)
            {
                JOptionPane.showMessageDialog(MyFrame,"No Card
ID","Alert",JOptionPane.ERROR_MESSAGE);
            }
        }catch(NumberFormatException f)
        {
            JOptionPane.showMessageDialog(MyFrame,"Number format
Exception","Alert",JOptionPane.ERROR_MESSAGE);
        }
    }
}

// function of withdrawl cash button

if (e.getSource() == WithdrawCashButton){
    try{
        //Get values from text fields
        String cardID12 = CardId3Text.getText();
        int CarddID1 = Integer.parseInt(cardID12);

        String PinNumber = PINNumber1Text.getText();
        int pinNumber1 = Integer.parseInt(PinNumber);

        String withdrawalAmount = WithdrawalAmountText.getText();
        int WithdrawalA = Integer.parseInt(withdrawalAmount);

        String year = (String)Year2List.getSelectedItem();
        String month = (String)Month2List.getSelectedItem();
        String day = (String)Day2List.getSelectedItem();
        String dateofwithdrawal = (year + month + day);

        if(Details.isEmpty()){
            // If no DebitCard is added, display an error message
            JOptionPane.showMessageDialog(MyFrame, "Cannot
Withdraw, DebitCard has not been added", "Alert",
JOptionPane.ERROR_MESSAGE);
        }
        else{
            for(BankCard withdrawCards : Details)
            {
                if(withdrawCards instanceof DebitCard){
                    if(CarddID1 == withdrawCards.getCardId()){
```

```
                if(pinNumber1 == ((DebitCard) withdrawCards)
.getPINnumber()){
                    if(WithdrawalA <= ((DebitCard)
withdrawCards).getBalanceAmount()){
                        // Withdraw the amount from the DebitCard
                        ((DebitCard)
withdrawCards).Withdraw(WithdrawalA, dateofwithdrawal, pinNumber1);
                        JOptionPane.showMessageDialog(MyFrame,
"The amount has been withdrawn successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    }
                    else{
                        // If withdrawal amount is greater than the
balance, display an error message
                        JOptionPane.showMessageDialog(MyFrame,
"Insufficient Balance", "Alert", JOptionPane.INFORMATION_MESSAGE);
                    }
                }
                else{
                    // If entered pin number is incorrect, display an error
message
                    JOptionPane.showMessageDialog(MyFrame,
"Incorrect Pin Number", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
            else{
                // If DebitCard with given ID is not found, display an
error message
```

```
                        JOptionPane.showMessageDialog(MyFrame, "
DebitCard with given Id has not been found", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                else{
                    // If DebitCard is not found, display an error message
                    JOptionPane.showMessageDialog(MyFrame, "DebitCard
NOT FOUND", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        }
    }
    catch(NumberFormatException nfe){
        JOptionPane.showMessageDialog(MyFrame, "The information
you provided can not be accepted", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}



// function of displaybutton
// This code block is executed when DisplayButton is clicked
if(e.getSource() == DisplayButton){
    // Check if there are any DebitCards added
    if(Details.isEmpty()){
```

```
                // Display an error message if there are no DebitCards
              JOptionPane.showMessageDialog(MyFrame, "Sorry, no
DebitCard has been added ", "Error", JOptionPane.ERROR_MESSAGE);
          }
          else{
               // Iterate through the list of BankCards to find the DebitCard and
display its details
              for(BankCard displaydebit: Details){
                 if(displaydebit instanceof DebitCard){
                    JOptionPane.showMessageDialog(MyFrame, "The details of
Debit Card has been displayed", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    System.out.println("\n");
                    ((DebitCard)displaydebit).display();
                    System.out.println("\n");
                 }
              }
          }
       }




       // Function to Clear Button
       if (e.getSource() == ClearButton)
```

```
        {
            CardIdText.setText(" ");
            CVCNumberText.setText(" ");
            clientnameText.setText(" ");
            IssuerBankText.setText(" ");
            BankAccountText.setText(" ");
            BalanceamountText.setText(" ");
            InterestRateText.setText(" ");
            CardId1Text.setText(" ");
            CreditLimitText.setText(" ");
            graceperiodText.setText(" ");
            CardId2Text.setText(" ");
            clientname1Text.setText(" ");
            IssuerBank1Text.setText(" ");
            BankAccount1Text.setText(" ");
            PINNumberText.setText(" ");
            Balanceamount1Text.setText(" ");
            CardId3Text.setText(" ");
            PINNumber1Text.setText(" ");
            WithdrawalAmountText.setText(" ");

        }

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

}
}
```