CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Unique Bajracharya

London Met ID: 22067577

College ID: NP01CP4A220074

Group: C4

Assignment Due Date: Wednesday, May 10, 2023

Assignment Submission Date: Tuesday, May 9, 2023

Table of Contents

Table of Figures

Table of Tables

1. Introduction

This report examines the Programming course material for Module Code CS400N1. The weighting for this programming course is 30% of the weighted for the entire module. The goal of the project is to create a real-world situation and overcome real-world challenges using Java's object-oriented programming language.

Java is a robust, platform-independent, object-oriented programming language used to create software that runs on various devices such as desktops and servers. It is a computer application development platform. The computer compiles the java code so that it can operate on multiple platforms. Java is used to construct java applications for laptops, phones, and servers due to its speed, security, and dependability.

This Coursework contains the information about the user engagement. The project involves in creating a Graphical User Interface (GUI) for a real-world problem scenario for bank card from the previous coursework. The GUI was designed to provide a user-friendly interface for managing the bank card details. We also had to pay attention to the design principles of software development to ensure that the system is easy to maintain and modify. This knowledge also strengthened our interaction with the command prompt.

1. Tools used in this Assignment.

BlueJ

This assignment is related to the development of GUI for the system that stores the details of the Bank Card. Bluej was used for the completion of this program. Bluej is simple and easy IDE for java.

Draw.io

Draw.io was the software that was used for the development of class diagram of the system in this assignment. Draw.io is a software that helps to make diagram and charts and custom layouts.

2. Aim of the Assignment

The aim of the assignment is to create GUI for the system that stores the details of Bank Card in an Array List by adding a class to the project that was developed in the previous coursework. This class will contain the main method and will be tested using the command prompt. As a text editor and compiler, I have used BlueJ for the completion of the program.

3. Objectives of the Assignment

1. Build a class BankCardGUI
2. Writing a report of the program
3. Testing the program
4. Solving the errors

2. Class Diagram

A class diagram is a graphical depiction of a system's static structure, displaying classes, their attributes, and methods. It shows the connections and dependencies between classes to provide a visual overview of a system's structure. Developers can acquire a better understanding of the system's design and how its many components work together by producing a class diagram. This allows them to plan and design the system before moving forward with development more effectively. Finally, a class diagram serves as a system blueprint, assisting developers in creating software that is both efficient and effective. (Pedamkar, 2022)

1. Class Diagram of BankCard

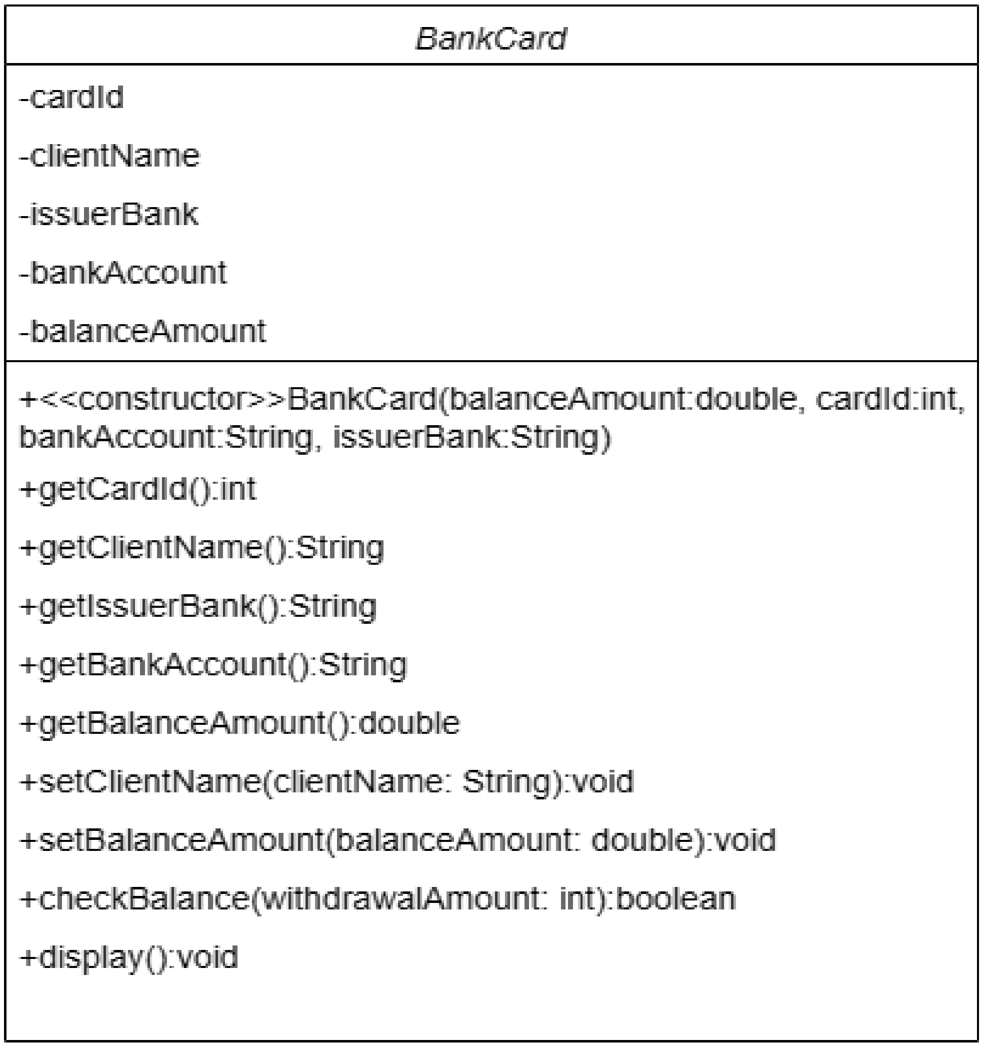

BankCard
-cardId
-clientName
-issuerBank
-bankAccount
-balanceAmount
+<<constructor>>BankCard(balanceAmount:double, cardId:int, bankAccount:String, issuerBank:String)
+getCardId():int
+getClientName():String
+getIssuerBank():String
+getBankAccount():String
+getBalanceAmount():double
+setClientName(clientName: String):void
+setBalanceAmount(balanceAmount: double):void
+checkBalance(withdrawalAmount: int):boolean
+display():void

Figure 1: Class Diagram for class BankCard

2. Class Diagram of DebitCard

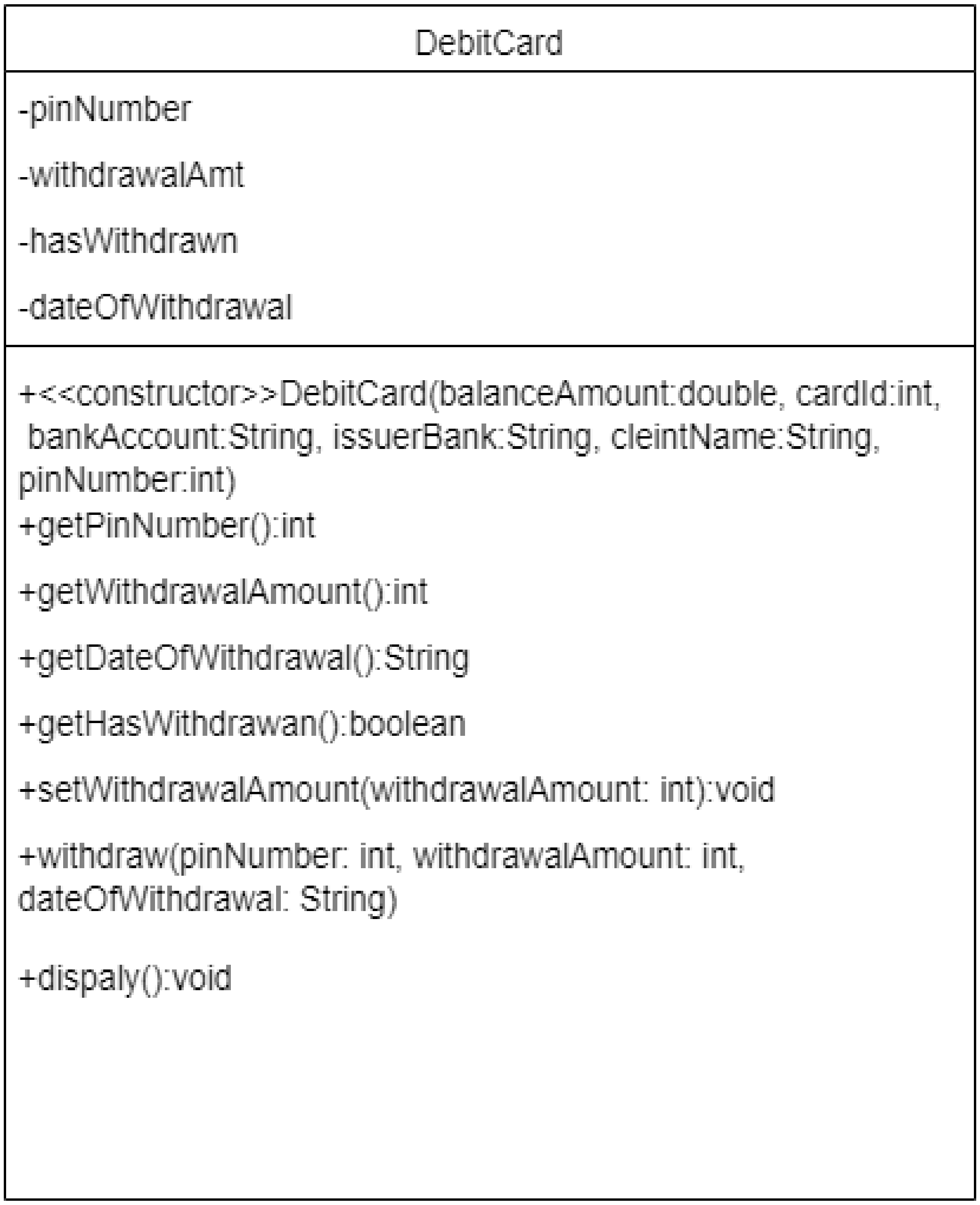
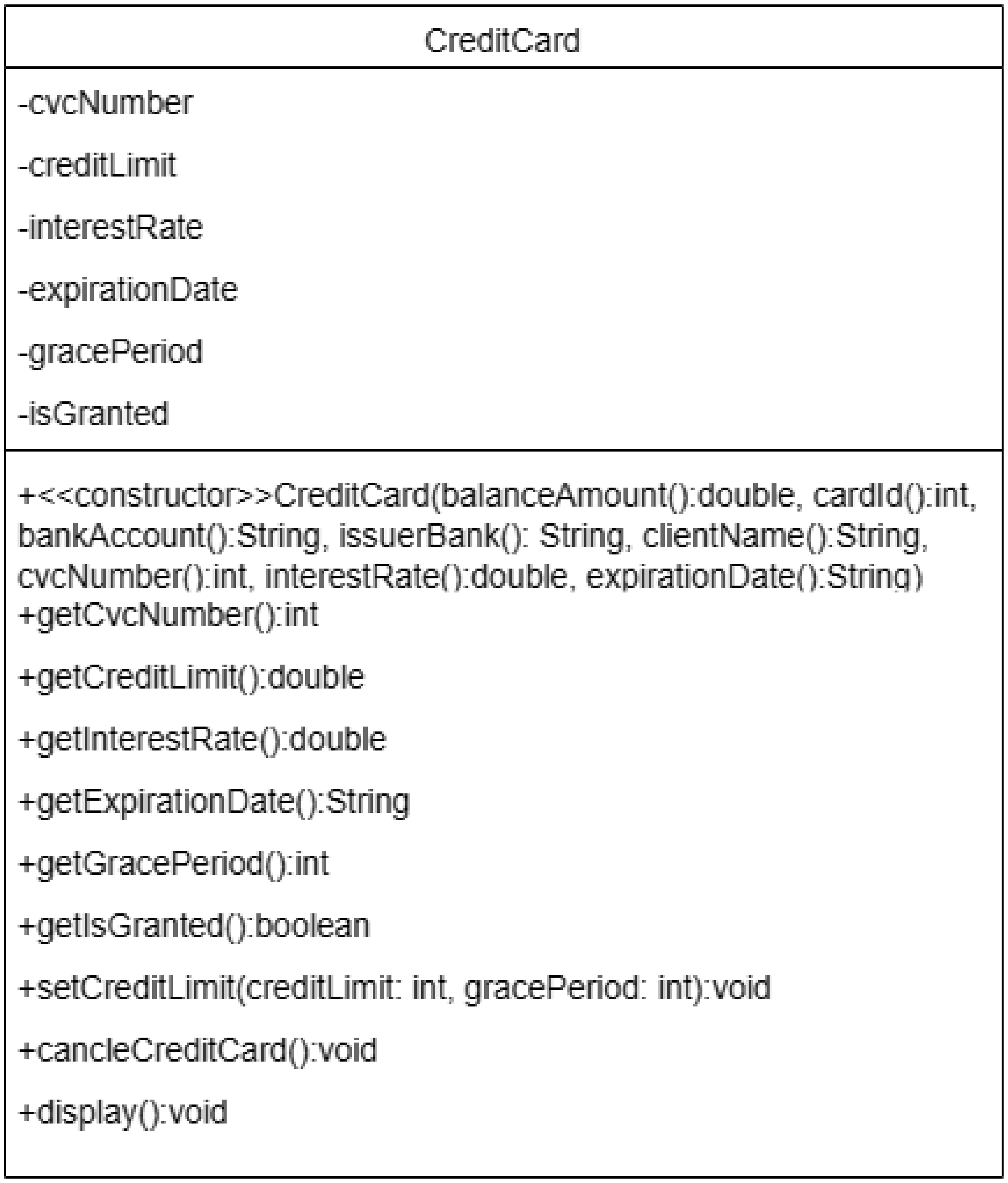

DebitCard
-pinNumber
-withdrawalAmt
-hasWithdrawn
-dateOfWithdrawal
+<<constructor>>DebitCard(balanceAmount:double, cardId:int, bankAccount:String, issuerBank:String, cleintName:String, pinNumber:int)
+getPinNumber():int
+getWithdrawalAmount():int
+getDateOfWithdrawal():String
+getHasWithdrawan():boolean
+setWithdrawalAmount(withdrawalAmount: int):void
+withdraw(pinNumber: int, withdrawalAmount: int, dateOfWithdrawal: String)
+dispaly():void

Figure 2: Class Diagram for class DebitCard

3. Class Diagram of CreditCard


CreditCard

-cvcNumber

-creditLimit

-interestRate

-expirationDate

-gracePeriod

-isGranted

+<<constructor>>CreditCard(balanceAmount():double, cardId():int, bankAccount():String, issuerBank(): String, clientName():String, cvcNumber():int, interestRate():double, expirationDate():String)

+getCvcNumber():int

+getCreditLimit():double

+getInterestRate():double

+getExpirationDate():String

+getGracePeriod():int

+getIsGranted():boolean

+setCreditLimit(creditLimit: int, gracePeriod: int):void

+cancleCreditCard():void

+display():void

Figure 3: Class Diagram for class CreditCard

4. Class Diagram for the System

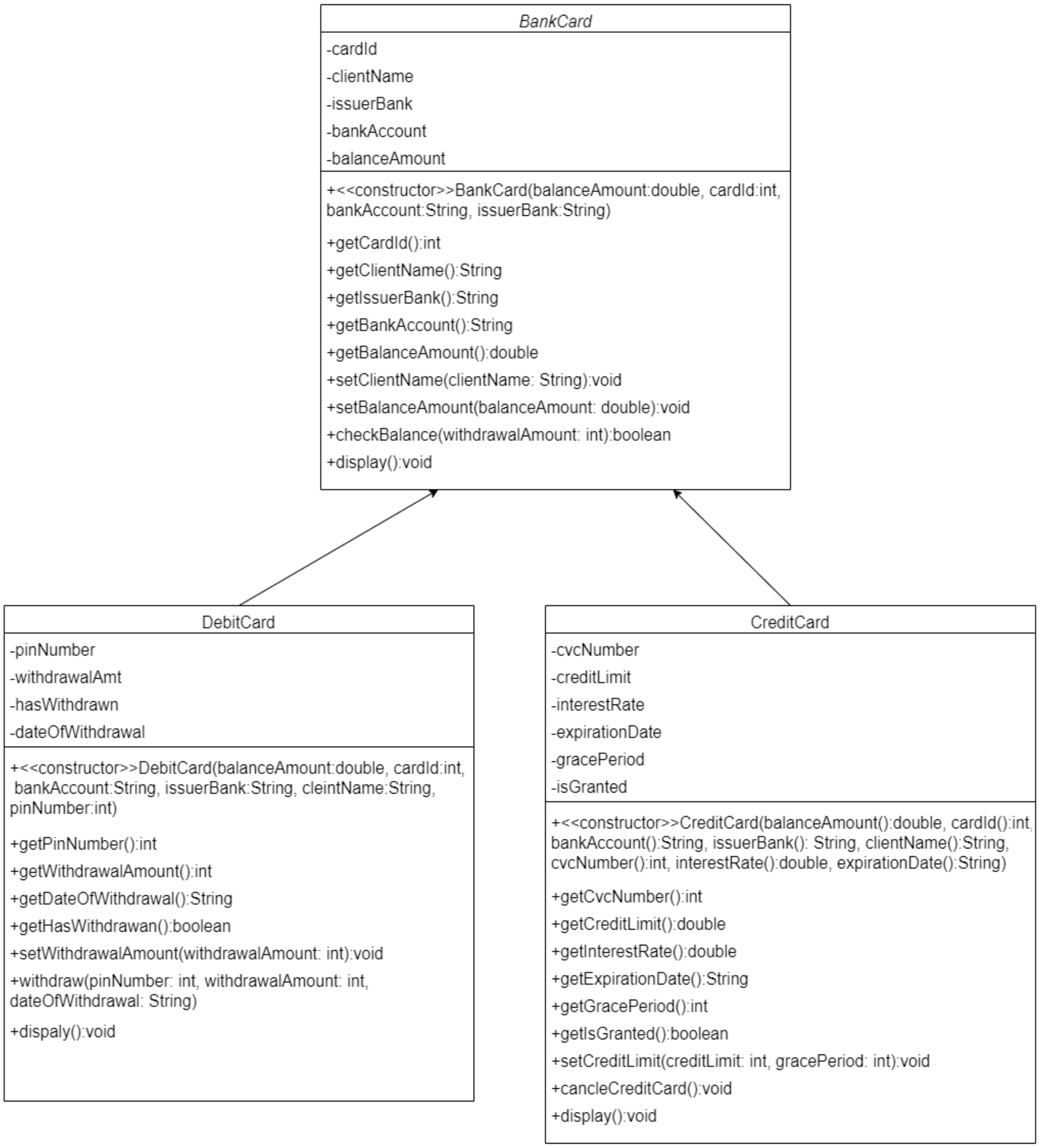
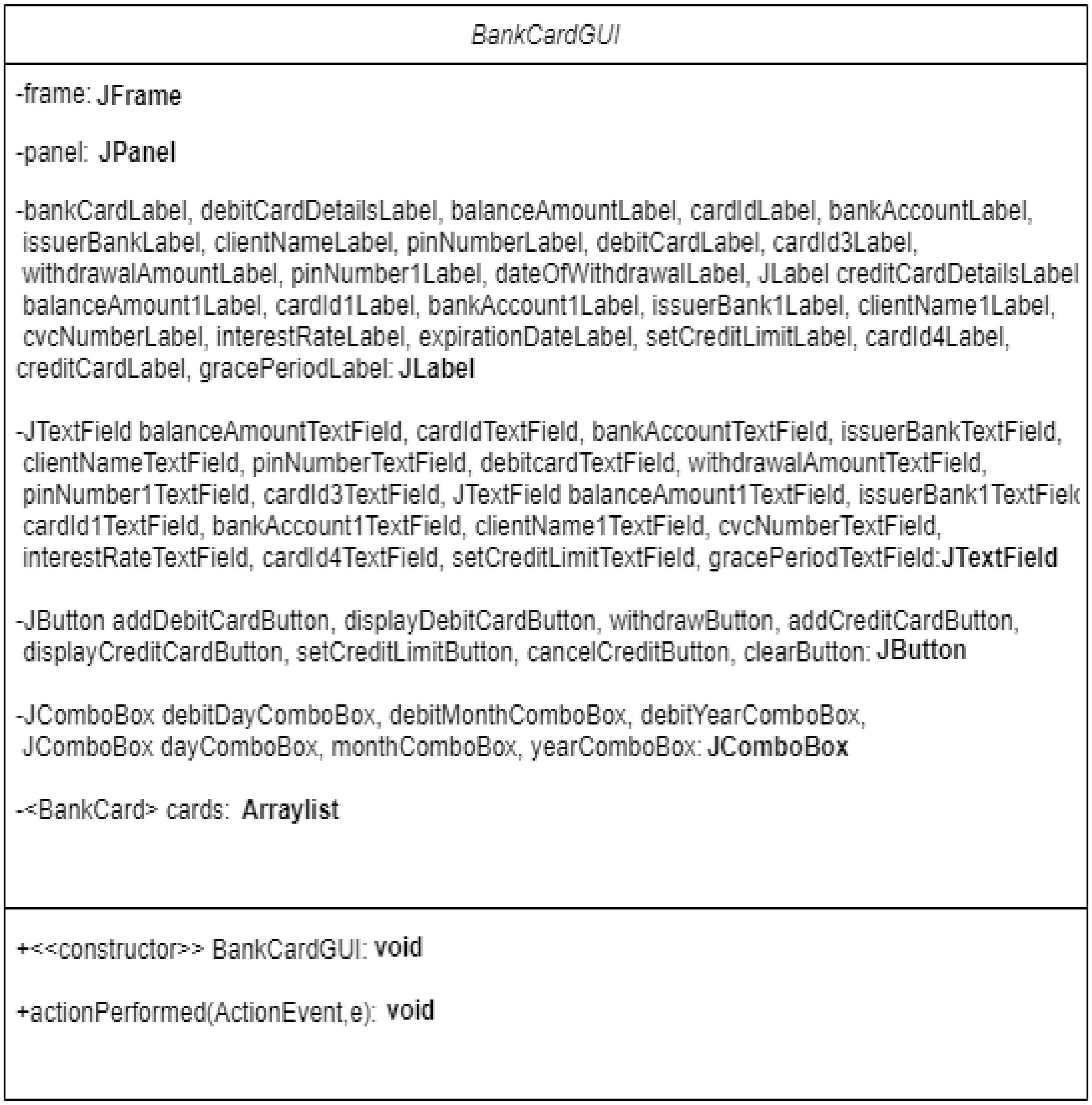

BankCard
-cardId
-clientName
-issuerBank
-bankAccount
-balanceAmount
+<<constructor>>BankCard(balanceAmount:double, cardId:int, bankAccount:String, issuerBank:String)
+getCardId():int
+getClientName():String
+getIssuerBank():String
+getBankAccount():String
+getBalanceAmount():double
+setClientName(clientName: String):void
+setBalanceAmount(balanceAmount: double):void
+checkBalance(withdrawalAmount: int):boolean
+display():void
DebitCard
-pinNumber
-withdrawalAmt
-hasWithdrawn
-dateOfWithdrawal
+<<constructor>>DebitCard(balanceAmount:double, cardId:int, bankAccount:String, issuerBank:String, cleintName:String, pinNumber:int)
+getPinNumber():int
+getWithdrawalAmount():int
+getDateOfWithdrawal():String
+getHasWithdrawan():boolean
+setWithdrawalAmount(withdrawalAmount: int):void
+withdraw(pinNumber: int, withdrawalAmount: int, dateOfWithdrawal: String)
+dispaly():void
CreditCard
-cvcNumber
-creditLimit
-interestRate
-expirationDate
-gracePeriod
-isGranted
+<<constructor>>CreditCard(balanceAmount():double, cardId():int, bankAccount():String, issuerBank(): String, clientName():String, cvcNumber():int, interestRate():double, expirationDate():String)
+getCvcNumber():int
+getCreditLimit():double
+getInterestRate():double
+getExpirationDate():String
+getGracePeriod():int
+getIsGranted():boolean
+setCreditLimit(creditLimit: int, gracePeriod: int):void
+cancleCreditCard():void
+display():void

Figure 4: Class Diagram for the System

5. Class Diagram of BankCardGUI



BankCardGUI

-frame: **JFrame**

-panel: **JPanel**

-bankCardLabel, debitCardDetailsLabel, balanceAmountLabel, cardIdLabel, bankAccountLabel, issuerBankLabel, clientNameLabel, pinNumberLabel, debitCardLabel, cardId3Label, withdrawalAmountLabel, pinNumber1Label, dateOfWithdrawalLabel, JLabel creditCardDetailsLabel balanceAmount1Label, cardId1Label, bankAccount1Label, issuerBank1Label, clientName1Label, cvcNumberLabel, interestRateLabel, expirationDateLabel, setCreditLimitLabel, cardId4Label, creditCardLabel, gracePeriodLabel: **JLabel**

-JTextField balanceAmountTextField, cardIdTextField, bankAccountTextField, issuerBankTextField, clientNameTextField, pinNumberTextField, debitcardTextField, withdrawalAmountTextField, pinNumber1TextField, cardId3TextField, JTextField balanceAmount1TextField, issuerBank1TextFiel cardId1TextField, bankAccount1TextField, clientName1TextField, cvcNumberTextField, interestRateTextField, cardId4TextField, setCreditLimitTextField, gracePeriodTextField:**JTextField**

-JButton addDebitCardButton, displayDebitCardButton, withdrawButton, addCreditCardButton, displayCreditCardButton, setCreditLimitButton, cancelCreditButton, clearButton: **JButton**

-JComboBox debitDayComboBox, debitMonthComboBox, debitYearComboBox, JComboBox dayComboBox, monthComboBox, yearComboBox: **JComboBox**

-<BankCard> cards: **Arraylist**

+<<constructor>> BankCardGUI: **void**

+actionPerformed(ActionEvent,e): **void**

Figure 5: Class Diagram of BankCardGUI

3. Pseudocode

Pseudocode is a simplified version of the actual code. It is not a true programming code, but rather a short phrase representation of each code before beginning to code a program in a specific programming language. It facilitates the construction of new programs and aids in making them understandable even to laymen. developing pseudocode before developing any program is an excellent habit that allows you to better arrange the code. (Airth, 2015)

1. Pseudocode of BankCardGUI

```
CREATE class BankCardGUI implementing ActionListener.

DO
DECLARE instance variable JFrame named frame

DECLARE instance variable JPanel named panel

DECLARE instance variable JLabel named after components of BankCard,
DebitCard and CreditCard

DECLARE instance variable JButton for adding DebitCard and CreditCard,
displaying DebitCard and Creditcard, withdrawing from DebitCard, setting the
credit Limit, cancelling the Creditcard and Clearing the details.

DECLARE instance variable JTextField named after components of BankCard,
DebitCard and CreditCard.
```

```
DECLARE instance variable JComboBox for expirationDate and withdrawal Date

DECLARE instance variable ArrayList of BankCard type named cards
END DO
```

#Pseudocode for building BankCardGUI Components

```
CREATE a constructor BankCardGUI

DO
CREATE arraylist of type BankCard named cards
END DO

DO
CREATE Jframe named frame
END DO

DO
CREATE JPanel named panel
SET layout of panel to null
SET background color of panel
SET bound for panel
```

```
ADD panel to javaframe
END DO

DO
CREATE labels named after components of BankCard
SET bounds for labels named after components of BankCard
ADD labels of components of BankCard in JavaPanel panel
END DO

DO
CREATE labels named after components of DebitCard
SET bounds for labels named after components of DebitCard
ADD labels of components of DebitCard in javaframe frame
END DO

DO
CREATE text fields named after components of DebitCard
SET bounds for text field named after components of DebitCard
ADD text fields of components of DebitCard in javaframe frame
END DO
```

```
DO
CREATE buttons for adding and Displaying details of DebitCard
SET bounds for the buttons for adding and displaying details of DebitCard
SET font for the buttons for adding and displaying details of DebitCard
SET background color for the buttons for adding and displaying details of DebitCard
SET foreground for the buttons
DISABLE focus painting for the buttons
SET size of buttons
ADD buttons for adding and displaying the details of the DebitCard
END DO

DO
CREATE comboBox for withdrawal date named debitDayComboBox, debitMonthComboBox, and debitYearComboBox
SET bounds for the comboBox for withdrawal date
ADD comboBox for withdrawal date
END DO

DO
CREATE buttons for withdrawing amount from DebitCard
SET bounds for withdrawing amount from DebitCard
SET font for the buttons for withdrawing amount of DebitCard
SET background color for the buttons for withdrawing amount of DebitCard
SET foreground for the buttons
DISABLE focus painting for the buttons
SET size of buttons
ADD buttons for adding and displaying the details of the DebitCard
END DO


DO
CREATE labels named after components of CreditCard
SET bounds for labels named after components of CreditCard
ADD labels of components of CreditCard in javaframe frame
END DO

DO
CREATE text fields named after components of CreditCard
SET bounds for text field named after components of CreditCard
ADD text fields of components of CreditCard in javaframe frame
END DO

DO
CREATE comboBox for expiration date named dayComboBox, monthComboBox,
and yearComboBox
```

```
SET bounds for the comboBox for expiration date
ADD comboBox for expiration date
END DO

DO
CREATE buttons for adding and Displaying details of CreditCard
SET bounds for the buttons for adding and displaying details of CreditCard
SET font for the buttons for adding and displaying details of CreditCard
SET background color for adding and displaying details of CreditCard
SET foreground for the buttons
DISABLE focus painting for the buttons
SET size of buttons
ADD buttons for adding and displaying the details of the CreditCard
END DO

DO
CREATE buttons for setting credit limit and cancelling credit
SET bounds for setting credit limit and cancelling credit
SET font for setting credit limit and cancelling credit
SET background color for setting credit limit and cancelling credit
SET foreground for the buttons
DISABLE focus painting for the buttons
```

```
SET size of buttons

ADD buttons for setting credit limit and cancelling credit

END DO


DO

SET frame size and set it’s visibility to true so that all the components would be visible

END DO


DO

ADD action listener for adding the debit card and credit card, displaying debit card and credit card, setting credit limit, cancelling credit card and clearing the details of debit and credit card.

END DO


CREATE main method
```

2. Pseudocode for BankCardGUI button functionality

```
CREATE method named actionPerformed with parameters(actionEvent e)

DO

        IF source for action is addDebitCard Button

                TRY
        GET the text for attributes of DebitCard
        CONVERT the invalid String type of input to integer
        CREATE a new CreditCard object with the extracted values
        ASSIGN creditCardDetails as CreditCard(balanceAmount1Value, cardId1Value, bankAccount, issuerBank, clientName, cvcNumberValue, interestRateValue, expirationDate)

        IF cards list is empty
            ADD debitCardDetails to cards list
            DISPLAY success message "DebitCard has been added"
        ELSE
            ASSIGN boolean isAdded as true
            ITERATE through cards list using a for-each loop
                IF cardId1Value equals card id in the array list
                    Display error message "DebitCard has already been added"
                    ASSIGN boolean isAdded to false
                    BREAK
                ELSE
                    ASSIGN boolean isAdded to true
                END IF
            END ITERATION
            IF Boolean isAdded is true
                ADD debitCardDetails to cards list
                DISPLAY success message "New DebitCard has been added"
            END IF
        END IF

    CATCH NumberFormatException
        DISPLAY warning message "The information you have input cannot be accepted"
    END CATCH
  END IF
END DO


DO
  IF source for action is addCreditCardButton
    TRY
        GET the text for attributes of DebitCard
        CONVERT the invalid String type of input to integer
        CREATE a new CreditCard object with the extracted values
        ASSIGN creditCardDetails as CreditCard(balanceAmount1Value, cardId1Value, bankAccount, issuerBank, clientName, cvcNumberValue, interestRateValue, expirationDate)

        IF cards list is empty
            ADD creditCardDetails to cards list
            DISPLAY success message "CreditCard has been added"
        ELSE
            ASSIGN Boolean isAdded as false
            Iterate through cards list using a for-each loop
                IF cardId1Value equals card id in the array list
                    DISPLAY error message "CreditCard has already been added"
                    ASSIGN boolean isAdded to false
                    BREAK
                ELSE
                    ASSIGN boolean isAdded to true
                END IF
            END ITERATION

            IF Boolean isAdded is true
                ADD creditCardDetails to cards list
```

```
                    DISPLAY success message "New CreditCard has been added"
                END IF
            END IF

        CATCH NumberFormatException
            DISPLAY warning message "The information you have input cannot be accepted"
        END CATCH

    END IF
END DO

DO
    IF source for action is displayDebitCardButton
        ASSIGN Boolean foundDebitCard as false

        ITERATE through cards list using a for-each loop
            IF storeDebitCards is an instance of DebitCard
                PRINT a new line
                CALL display() method on (DebitCard) storeDebitCards
                PRINT a new line
                **ASSIGN** Boolean foundDebitCard as true
            **END IF**
        **END ITERATION**

        **IF** Boolean foundDebitCard is true
            **DISPLAY** information message "The details have been displayed"
        **ELSE**
            **DISPLAY** information message "No DebitCard found"
        **END IF**

    **END IF**
**END DO**

**DO**
    **IF** source for action is displayCreditCardButton
        **ASSIGN** Boolean foundCreditCard as false

        **ITERATE** through cards list using a for-each loop
            **IF** storeCreditCards is an instance of CreditCard
                **PRINT** a new line
                **CALL** display() method on (CreditCard) storeCreditCards
                **PRINT** a new line
                **ASSIGN** Boolean foundCreditCard as true
            **END IF**
        **END ITERATION**

        **IF** Boolean foundCreditCard is true
            **DISPLAY** information message "The details have been
    displayed"
        **ELSE**
            **DISPLAY** information message "No CreditCard found"
        **END IF**
    **END IF**
**END DO**


**DO**
    **IF** source for action is withdrawButton
        **TRY**
            **GET** the text for attributes of DebitCard
            **CONVERT** the invalid String type of input to integer
            **ASSIGN** Boolean debitcard as false

            **IF** cards list is empty
```

```
                  DISPLAY error message "Cannot Withdraw, DebitCard has not been added yet"
              ELSE
                  ITERATE through cards list using a for-each loop
                      IF withdrawCards is an instance of DebitCard
                          IF cardId3Value equals card id in the arraylist
                              ASSIGN Boolean debitcard as true

                              IF pinNumberWithdrawValue equals pin number in the array list
                                  IF withdrawal amount is less or equals to the balance amount
                                      CALL withdraw method with (pinNumberWithdrawValue, withdrawalValue, dateOfWithdrawal) as parameter
                                      DISPLAY success message "The amount has been withdrawn successfully"
                                      Break the loop
                                  ELSE
                                      DISPLAY information message "Insufficient balance"
                                  END IF
                              ELSE
```

```
                                    DISPLAY error message "Incorrect PIN number"
                                END IF
                            ELSE
                                ASSIGN debitcard as false
                            END IF
                        ELSE
                            DISPLAY error message "Debit card not found"
                        END IF
                    END ITERATION

                END IF

                IF Boolean debitcard is false
                    DISPLAY error message "The DebitCard with the provided ID has not been found"
                END IF

            CATCH NumberFormatException
                DISPLAY warning message "The information you provided cannot be accepted"

            END CATCH
```

```
        END IF
END DO


DO
        IF source for action is setCreditLimitButton
                TRY
                        GET the text for attributes of DebitCard
                        CONVERT the invalid String type of input to integer
                        ASSIGN Boolean creditcard as false

                        IF cards list is empty
                                DISPLAY error message "Cannot set credit limit. CreditCard has not been added yet"
                        ELSE
                                Iterate through cards list using a for-each loop
                                        IF creditCards is an instance of CreditCard
                                                IF cardIdCreditLimit equals creditCards.getCardId()
                                                        ASSIGN Boolean creditcard as true
                        IF setCreditLimitValue is less than or equal to 2.5 times the balance amount
                            CALL setCreditLimit method and pass (setCreditLimitValue, gracePeriodValue) as parameters
                            DISPLAY success message "The credit limit has been set successfully"
                            BREAK the loop
                        ELSE
                            DISPLAY error message "The amount exceeds the credit limit"
                        END IF
                    ELSE
                        ASSIGN Boolean creditcard as false
                    END IF
                END IF
            END ITERATION

        END IF

        IF Boolean creditcard is false
            DISPLAY error message "The Creditcard with the provided ID has not been found"
        END IF
        CATCH NumberFormatException
            DISPLAY warning message "The information you provided
    cannot be accepted"

        END CATCH

    END IF
END DO

DO
    IF source for action is cancelCreditButton
        TRY
            GET the text for attributes of DebitCard
            CONVERT the invalid String type of input to integer
            ASSIGN Boolean creditlimit as false

            IF cards list is empty
                DISPLAY error message "Cannot cancel credit card.
    CreditCard has not been added yet"
            ELSE
                Iterate through cards list using a for-each loop
                    IF creditCards is an instance of CreditCard
                    IF cardIdCreditLimit equals the card id in the array list
                        CALL cancelCreditCard() method
                        DISPLAY success message "The CreditCard has been canceled"
                        ASSIGN boolean creditlimit as true
                        BREAK the loop
                    ELSE
                        DISPLAY error message "The card ID provided does not exist"
                    END IF
                ELSE
                    ASSIGN Boolean creditlimit as false
                END IF
            END ITERATION
        END IF

        IF Boolean creditlimit is false
            DISPLAY message "CreditCard not found."
        END IF

    CATCH NumberFormatException
```

```
                **DISPLAY** warning message "The information you provided cannot be accepted"

            **END CATCH**

        **END IF**

    **END DO**

    **DO**

        **IF** source for action is clearButton

            **SET** balanceAmountTextField text to empty string
            **SET** cardIdTextField text to empty string
            **SET** bankAccountTextField text to empty string
            **SET** issuerBankTextField text to empty string
            **SET** clientNameTextField text to empty string
            **SET** pinNumberTextField text to empty string
            **SET** balanceAmount1TextField text to empty string
            **SET** cardId1TextField text to empty string
            **SET** bankAccount1TextField text to empty string
            **SET** issuerBank1TextField text to empty string
            **SET** clientName1TextField text to empty string
            SET cvcNumberTextField text to empty string
            SET interestRateTextField text to empty string
            SET pinNumber1TextField text to empty string
            SET cardId3TextField text to empty string
            SET withdrawalAmountTextField text to empty string
            SET pinNumberTextField text to empty string
            SET cardId4TextField text to empty string
            SET setCreditLimitTextField text to empty string
            SET gracePeriodTextField text to empty string
        END IF
END DO
```

4. Method Description of BankCardGUI

1. Add button for DebitCard

When the ADD button in the GUI is pressed, multiple variables are initialized, and the user-inputted values are assigned to them. An error message is presented instantly if any required fields are left blank to ensure that all required fields are filled. The program then iterates over the arraylist, comparing the card id from the list to the one entered by the user. If they are identical, an error message indicating that the card already exists is displayed. If the card id is not found in the arraylist, the variables are provided to the DebitCard class's constructor as an object of the same class. After that, the newly formed object is added to the arraylist. All these processes are carried out within a try block to enable the handling of any potential exceptions with a catch block.

2. Add button for CreditCard

When the ADD button in the GUI is pressed, various variables are initialized, and the values entered by the user are allocated to them. If the fields are left blank, an error message is presented to the user instantly to ensure that all requited fields are filled. The program then iterates over the array list, to compare the card id from the list to the one provided by the user. If any of the id are matched, an error message indicating that the card already exists. If the card is not to be found in the array list, the variables are then provided to the CreditCard constructor as an object of the same class. After that, the newly formed object is added to the arraylist. All these processes are carried out within a try block for handling any potential exceptions.

3. Display button for DebitCard

When the user presses the "Display DebitCard" button in the GUI, the BankCardGUI invokes the DebitCard class's display function. It then iterates over the arraylist, displaying all the data of the client's DebitCard with a specified card id on the screen. And a display popup message is displayed.

4. **Display button for CreditCard**

When a user presses the "Display CreditCard" button on the GUI, the BankCardGUI invokes the CreditCard class's display function. The software then loops over the arraylist, display a popup message and prints out on the screen all of the information related to the client's CreditCards.

5. **Withdraw button for DebitCard**

When the user presses the “WithDraw” button on the GUI, the BankCardGUI calls the DebitCard class’s withdraw method. When the withdraw button is pressed, various variable are initialized, and values entered by the user are allocated to them. If any required fields are left out, an error message is displayed asking the user to fill them out. The card id provide by the user is then compared with the card id present in the array list and also checks if the inputed pin number is same as the pin number of the specific card id in the array list, after the arraylist has been iterated. It also checks if the balance amount is sufficient for withdrawal by calling the checkBalance method of the BankCard class. Then the BankCardGUI class calls the withdraw method of the DebitCard class. Else an error message is displayed to the user.

6. **Set Credit Limit button for CreditCard**

When the user presses the “set credit limit” button on the GUI, the BankCardGUI calls the CreditCard class’s setCreditLimit method. When the Set Credit Limit button is presses, various variable are initialized, and values entered by the user is allocated to them. If any mandatory fields are left blank, an error message is displayed asking the user to add the creditcard first. The card id provided by the user matches the card id in the array list while comparing them after the arraylist has been iterated. If they are equal, it again checks if the credit limit is less than or equals to 2.5 times the balance amount. Then the BankCardGUI class calls the setCreditLimit method from the CreditCard class. Else an error message is displayed to the user. All these operations are carried out in a catch block for any exception handling if arised.

7. Cancel CreditCard

When the cancel credit button is pressed, the BankCardGUI calls the CreditCard class’s cancelCreditCard method. When the Cancel CreditCard button is presses, various variable is initialized, and values entered by the user is allocated to them. If any mandatory fields are left blank, an error message is displayed asking the user to add the creditcard first. The card id provided by the user matches the card id in the array list while comparing them after the arraylist has been iterated. Then all the details of the CreditCard are set to default. Else an error message is displayed to the user. All these operations are carried out in a catch block for any type of exception handling.

8. Clear Button

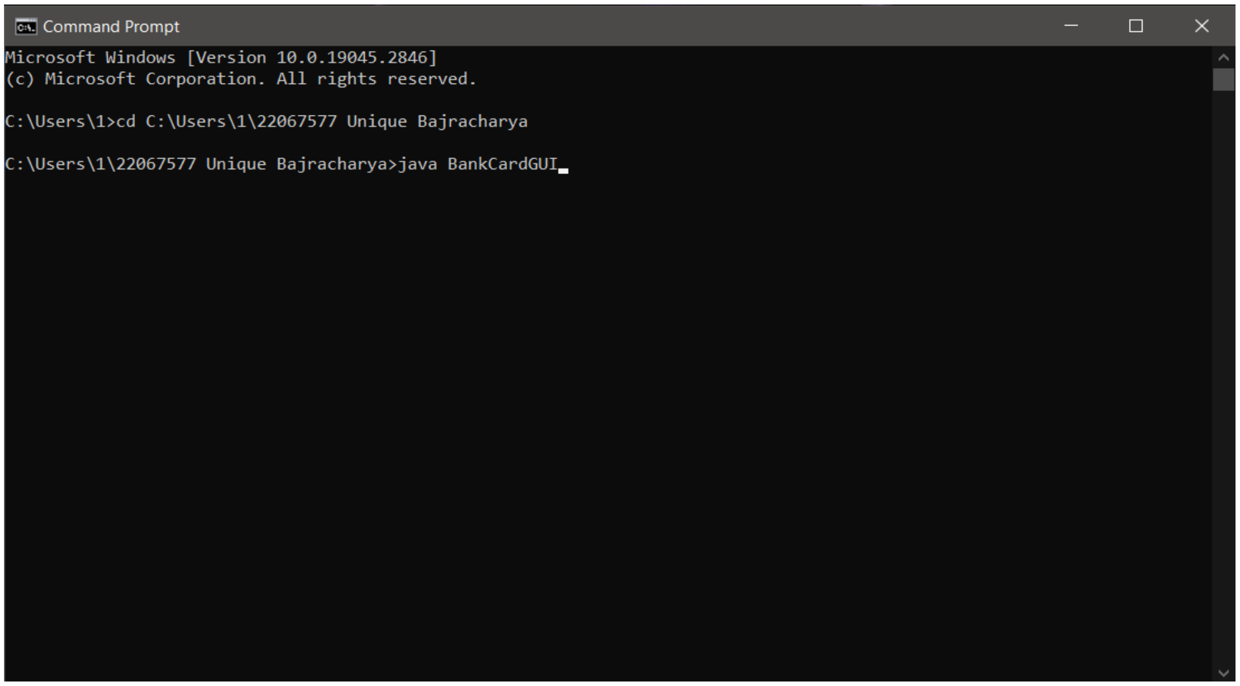
When the "CLEAR" button is pushed, the values of all text fields in the TransportGUI GUI are set to default.

5. Testing (Inspection)

1. Test - 1

Test No.:	1
Objective:	Test that the program can be compiled and run using the command prompt
Action:	• The path to the Java package we need to use is entered into the command prompt. • The command “java Filename” is used to run the BankCardGUI class.
Expected Result:	A java frame should be opened.
Actual Result:	A java frame was opened.
Conclusion:	The test is successful.

Table 1: Test Table for running the program from command prompt



```
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\1>cd C:\Users\1\22067577 Unique Bajracharya

C:\Users\1\22067577 Unique Bajracharya>java BankCardGUI
```

Figure 6: Running the program using command prompt

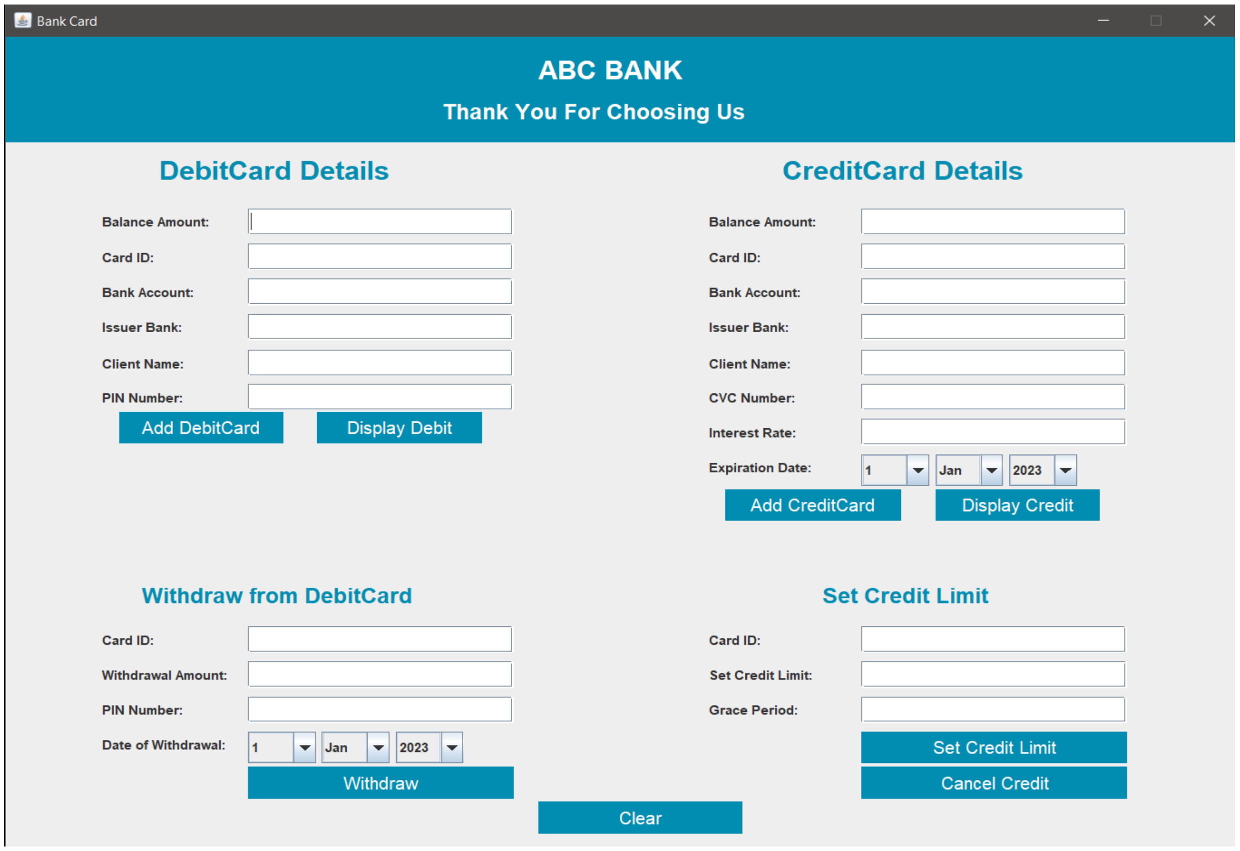


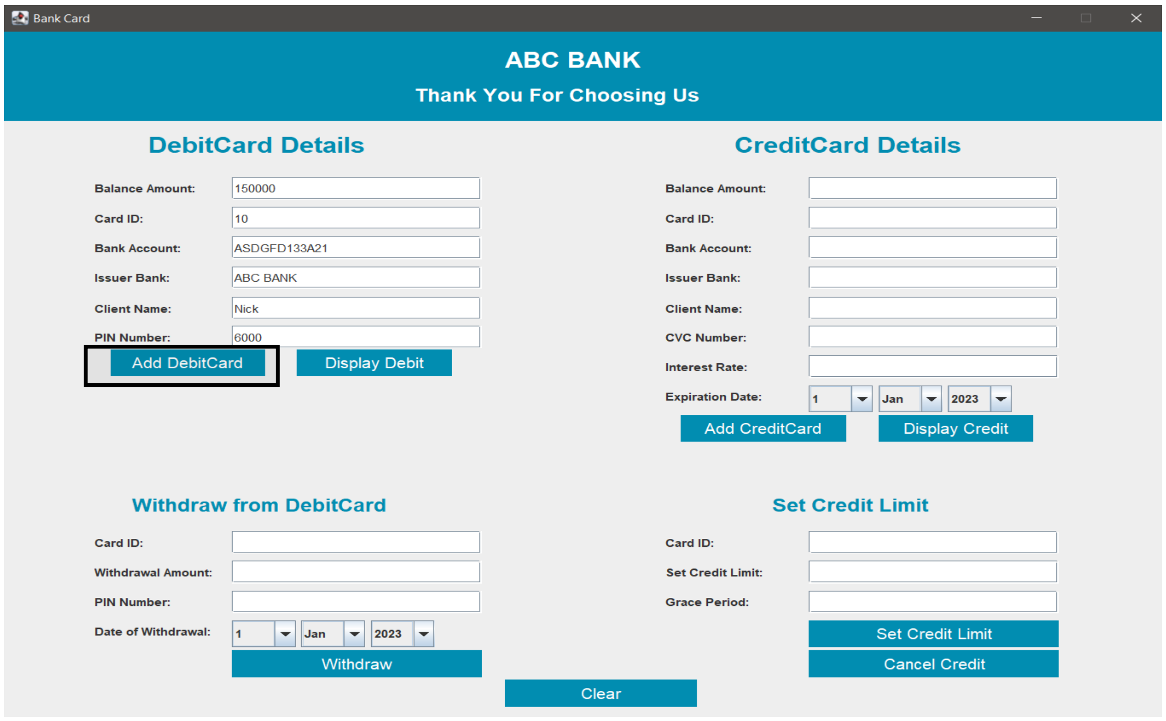
Figure 7:GUI opened after running the program from command prompt

2. Test – 2

a. Test - a (Add button for DebitCard)

Test No.:	a
Objective:	Add button for Debit Card
Action:	• After running the program DebitCard is passed in the GUI. • The DebitCard is added using the following values: Balance amount = 150000 Card id = 10 Bank Account = ASDGFD133A21 Issuer Bank = ABC BANK Client Name = Nick PIN Number = 6000 • "Add DebitCard" button is pressed
Expected Result:	DebitCard should be added and a information message should pop up.
Actual Result:	DebitCard was added and a information message was popped up.
Conclusion:	The test is successful.

Table 2: Test Table to add Debit Card in BankCardGUI

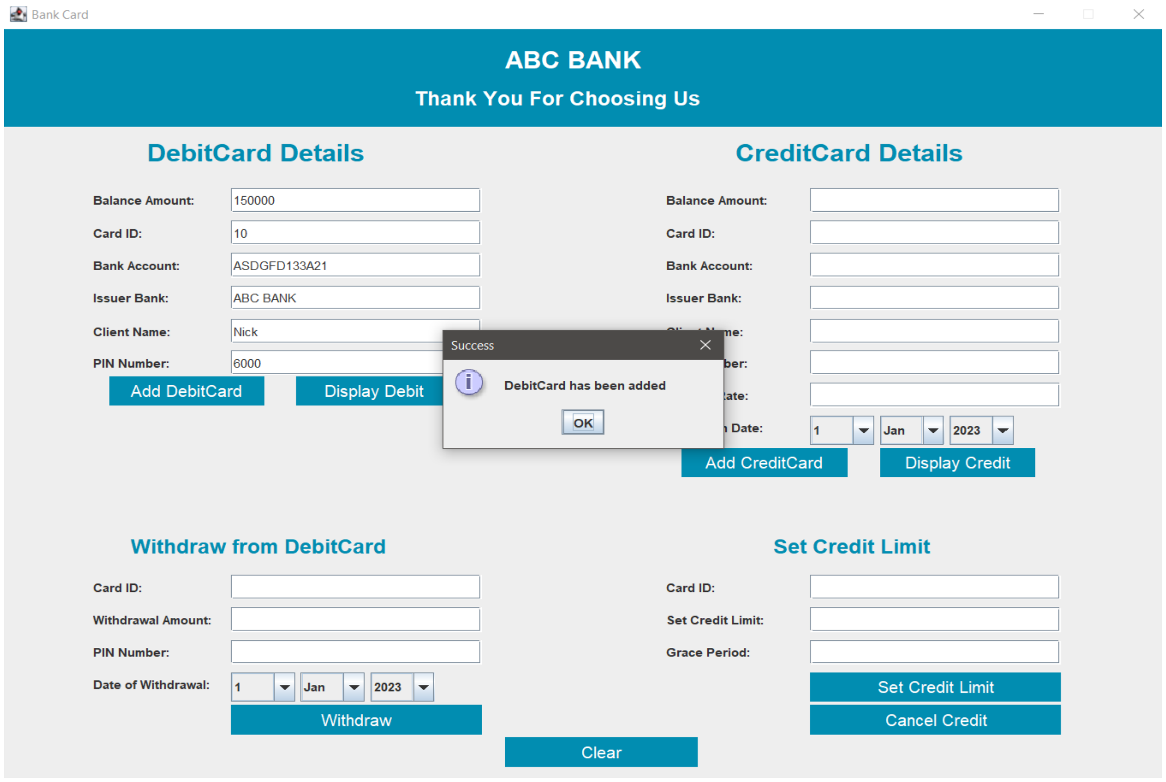


Balance Amount:	150000
Card ID:	10
Bank Account:	ASDGFD133A21
Issuer Bank:	ABC BANK
Client Name:	Nick
PIN Number:	6000

Expiration Date: 1 Jan 2023

Date of Withdrawal: 1 Jan 2023

Figure 8: Adding DebitCard to the Arraylist



Balance Amount:	150000
Card ID:	10
Bank Account:	ASDGFD133A21
Issuer Bank:	ABC BANK
Client Name:	Nick
PIN Number:	6000

DebitCard has been added

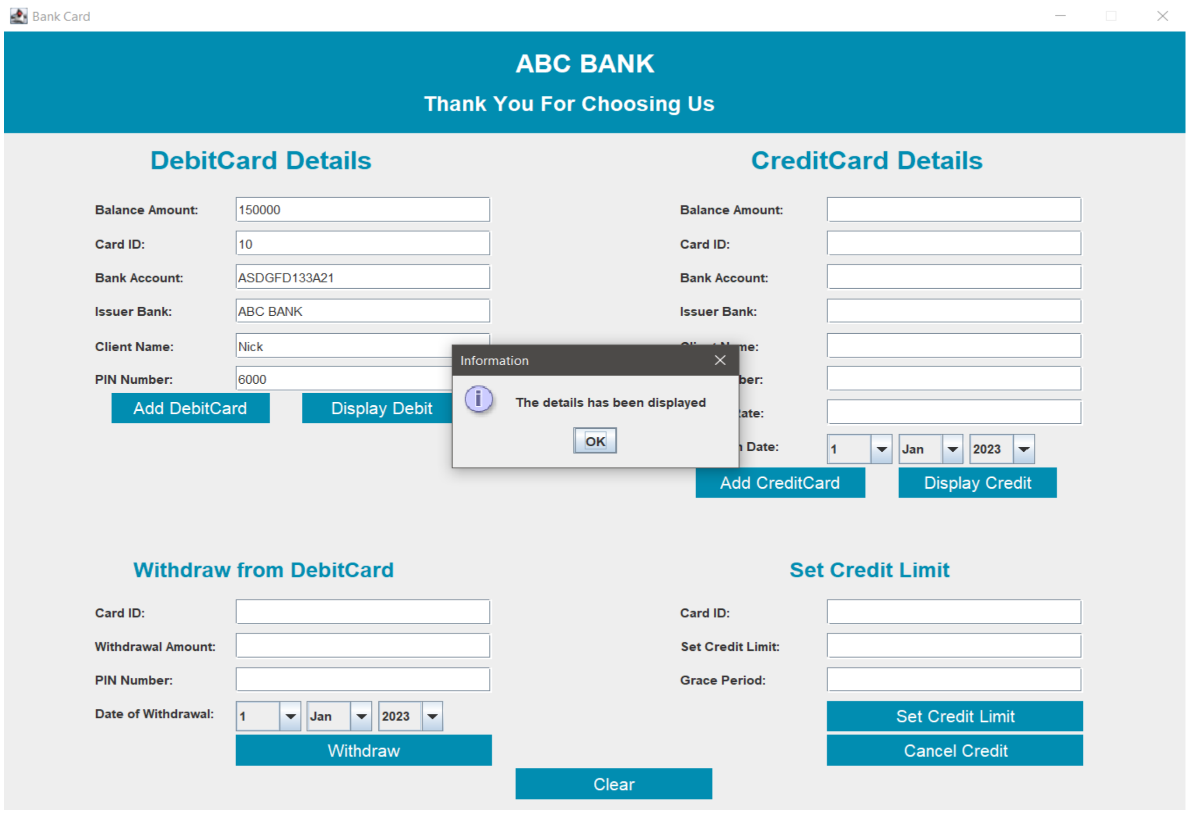
Date of Withdrawal: 1 Jan 2023

Figure 9: Success Message for Adding DebitCard to the Arraylist

b. Test - b

Test No.:	b
Objective:	Display button for DebitCard
Action:	• After running the program CreditCard is passed in the GUI. • “Display DebitCard” button is pressed
Expected Result:	Details of DebitcCard should be displayed.
Actual Result:	Details of DebitCard is displayed.
Conclusion:	The test is successful.

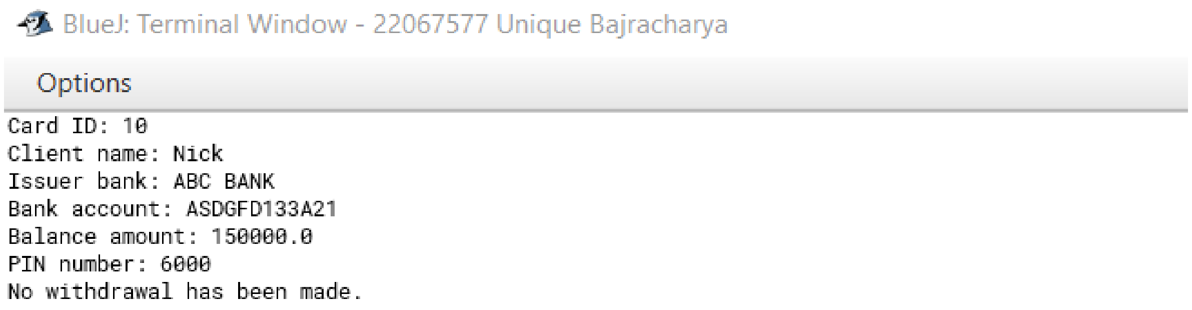
Table 3: Test Table for Display button of DebitCard in BankCardGUI



Balance Amount:	150000
Card ID:	10
Bank Account:	ASDGFD133A21
Issuer Bank:	ABC BANK
Client Name:	Nick
PIN Number:	6000

The details has been displayed

Figure 10: Display DebitCard Details



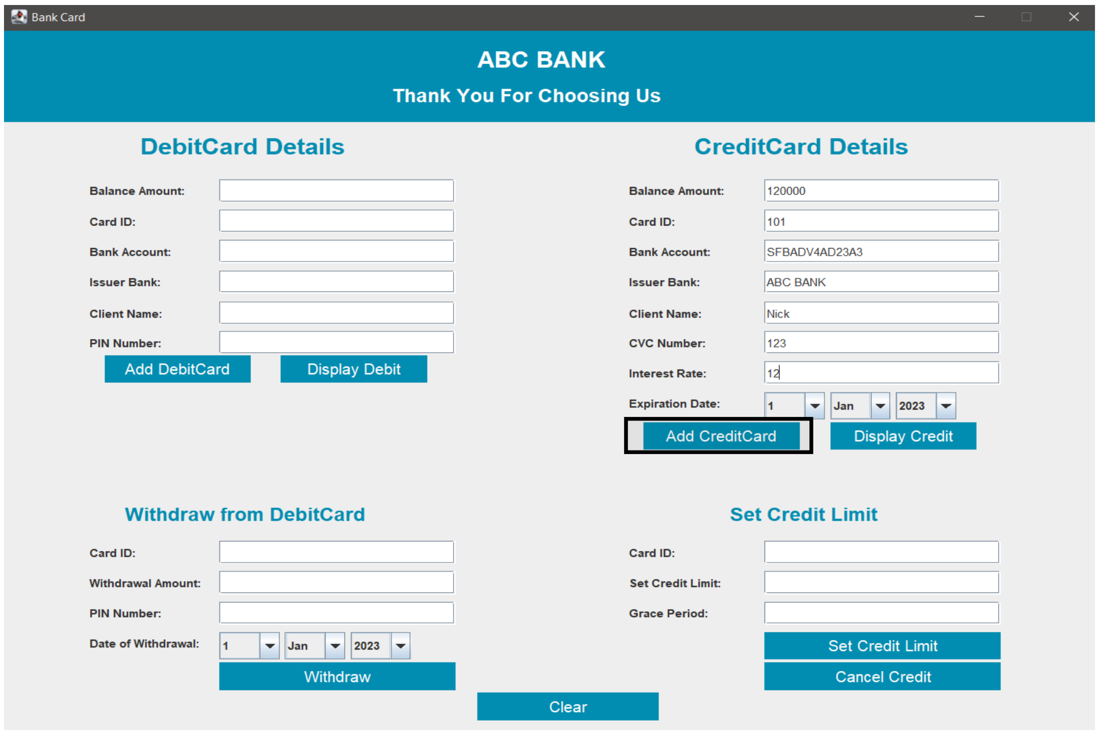
```
Card ID: 10
Client name: Nick
Issuer bank: ABC BANK
Bank account: ASDGFD133A21
Balance amount: 150000.0
PIN number: 6000
No withdrawal has been made.
```

Figure 11: DebitCard Details

c. Test - c (Add Button for CreditCard)

Test No.:	c
Objective:	Add button for CreditCard
Action:	• After running the program CreditCard is passed in the GUI. • The DebitCard is added using the following values: Balance amount = 120000 Card id = 101 Bank Account = SFBADV4AD23A3 Issuer Bank = ABC BANK Client Name = Nick CVC Number = 123 Interest Rate = 12 Expiration date = 1st jan 2023 • "Add CreditCard" button is pressed
Expected Result:	CreditCard should be added and a information message should pop up.
Actual Result:	CreditCard was added and a information message was popped up.
Conclusion:	The test is successful.

Table 4: Test Table to add Credit Card in BankCardGUI



Bank Card

ABC BANK

Thank You For Choosing Us

DebitCard Details

Balance Amount:	
Card ID:	
Bank Account:	
Issuer Bank:	
Client Name:	
PIN Number:	

CreditCard Details

Balance Amount:	120000
Card ID:	101
Bank Account:	SFBADV4AD23A3
Issuer Bank:	ABC BANK
Client Name:	Nick
CVC Number:	123
Interest Rate:	12
Expiration Date:	1 Jan 2023

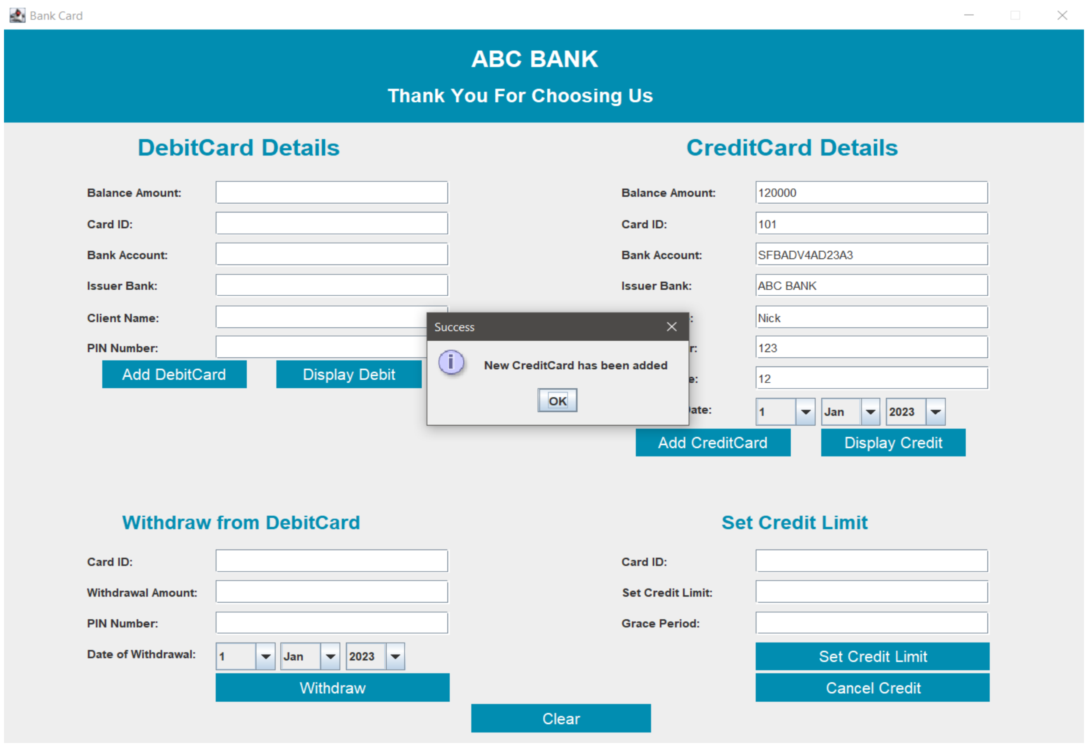
Withdraw from DebitCard

Card ID:	
Withdrawal Amount:	
PIN Number:	
Date of Withdrawal:	1 Jan 2023

Set Credit Limit

Card ID:	
Set Credit Limit:	
Grace Period:	

Figure 12: Adding CreditCard to the Arraylist



Bank Card

ABC BANK

Thank You For Choosing Us

DebitCard Details

CreditCard Details

Balance Amount:	120000
Card ID:	101
Bank Account:	SFBADV4AD23A3
Issuer Bank:	ABC BANK
	Nick
	123
	12
	1 Jan 2023

Success

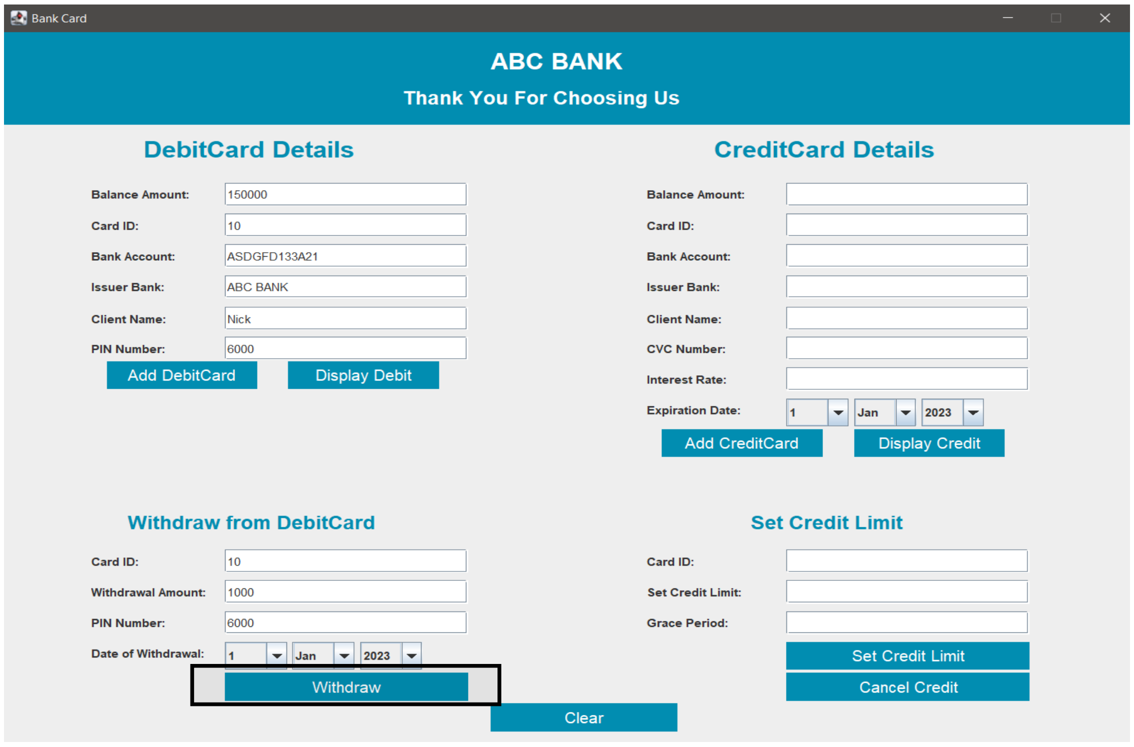
New CreditCard has been added

Figure 13: Success Message for Adding CreditCard to the Arraylist

d. Test – d (Withdraw amount from DebitCard)

Test No.:	d
Objective:	Withdraw amount from DebitCard
Action:	• After running the program, Amount is withdrawan from DebitCard. • The amount is withdrawn using the following values: Card id = 10 Withdrawal Amount = 1000 PIN Number = 6000 Date of Withdrawal = 1st jan 2023 • “Withdraw” button is pressed
Expected Result:	The amount should be withdrawan and a information message should pop up.
Actual Result:	The amount was withdrawan and a information message was popped up.
Conclusion:	The test is successful.

Table 5: Test Table to withdraw from Debit Card in BankCardGUI



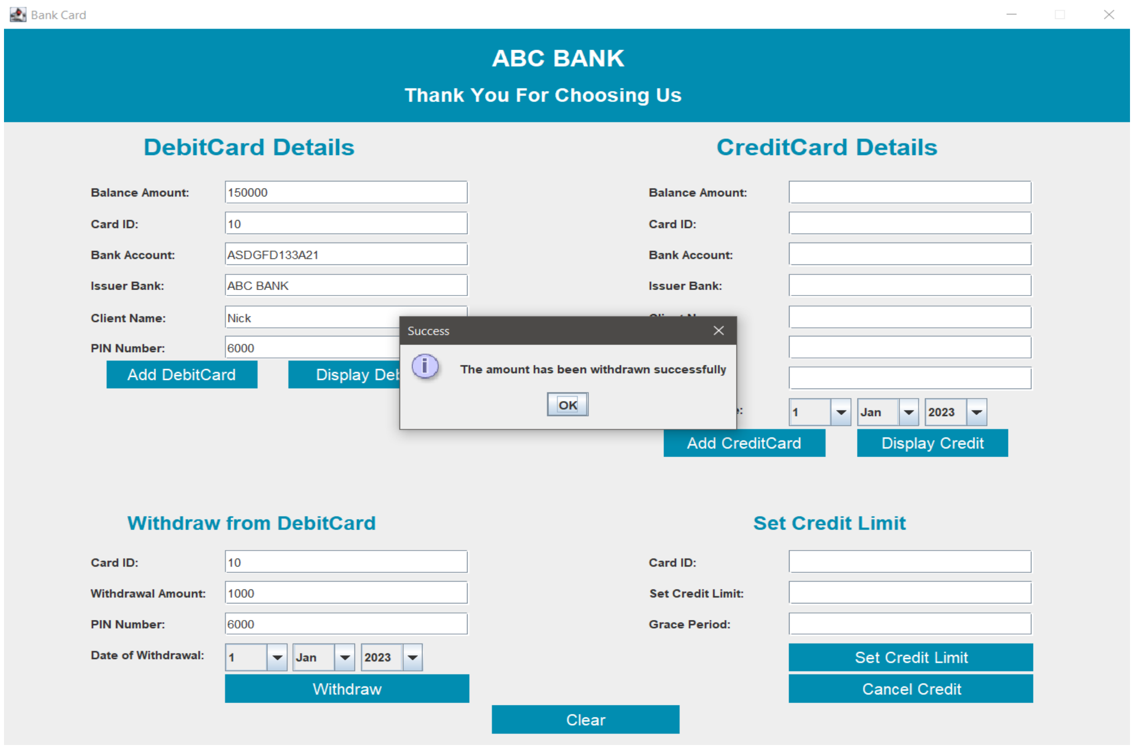
Balance Amount:	150000
Card ID:	10
Bank Account:	ASDGFD133A21
Issuer Bank:	ABC BANK
Client Name:	Nick
PIN Number:	6000

Balance Amount:	
Card ID:	
Bank Account:	
Issuer Bank:	
Client Name:	
CVC Number:	
Interest Rate:	
Expiration Date:	1 Jan 2023

Card ID:	10
Withdrawal Amount:	1000
PIN Number:	6000
Date of Withdrawal:	1 Jan 2023

Card ID:	
Set Credit Limit:	
Grace Period:	

Figure 14: Withdraw Amount

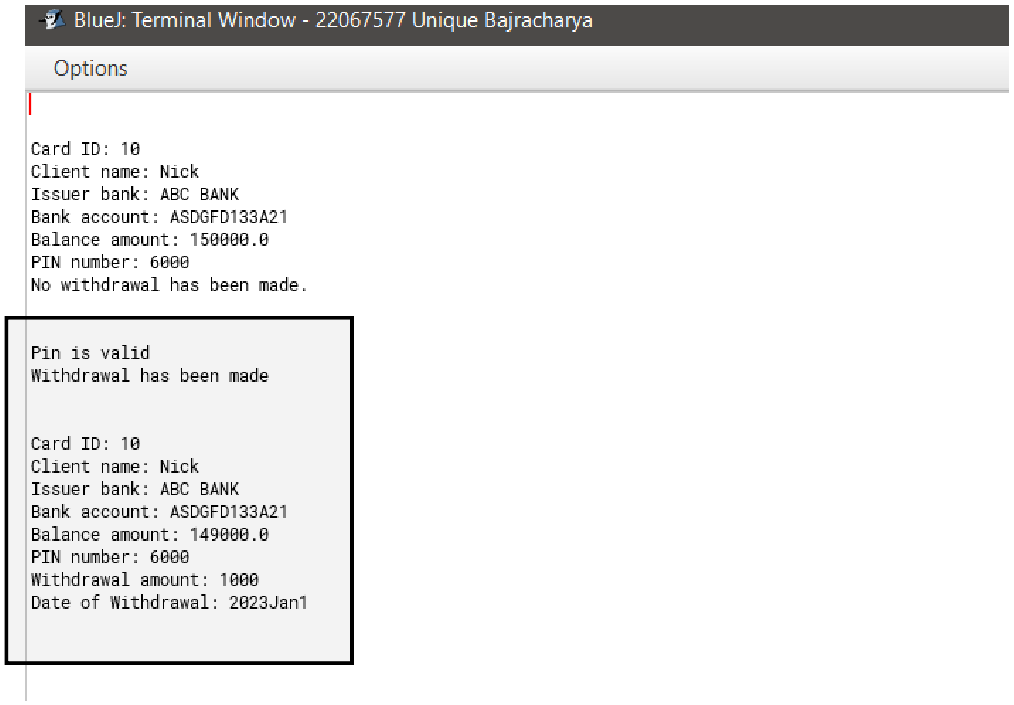


The amount has been withdrawn successfully

Balance Amount:	150000
Card ID:	10
Bank Account:	ASDGFD133A21
Issuer Bank:	ABC BANK
Client Name:	Nick
PIN Number:	6000

Card ID:	10
Withdrawal Amount:	1000
PIN Number:	6000
Date of Withdrawal:	1 Jan 2023

Figure 15: Success Message for Withdrawal



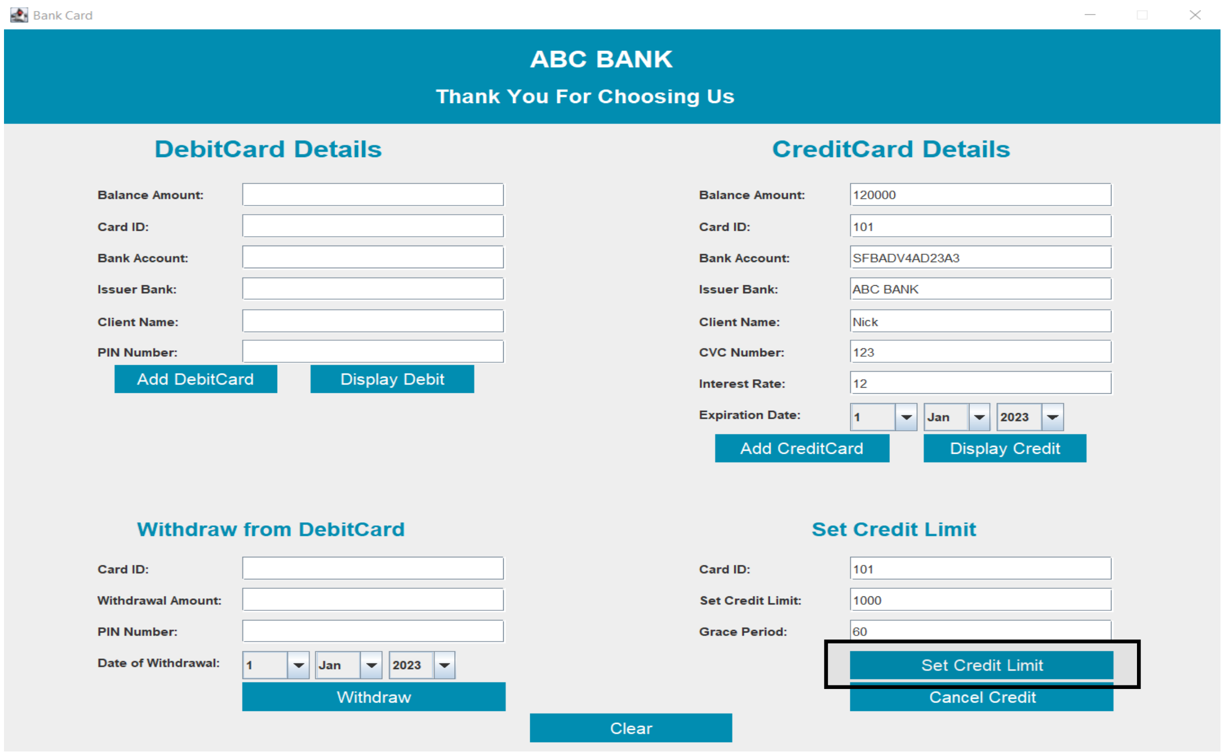
```
Card ID: 10
Client name: Nick
Issuer bank: ABC BANK
Bank account: ASDGFD133A21
Balance amount: 150000.0
PIN number: 6000
No withdrawal has been made.

Pin is valid
Withdrawal has been made


Card ID: 10
Client name: Nick
Issuer bank: ABC BANK
Bank account: ASDGFD133A21
Balance amount: 149000.0
PIN number: 6000
Withdrawal amount: 1000
Date of Withdrawal: 2023Jan1
```

Figure 16: Updated Display of DebitCard Details

e. Test - e

Test No.:	e
Objective:	Set credit limit for CreditCard
Action:	• After running the program, credit limit is set in the credit card. • The credit limit is set using following values: Card id = 101 Credit Limit = 1000 Grace period = 60 • “Set Credit Limit” button is pressed
Expected Result:	The credit limit should be set and a information message should pop up.
Actual Result:	The credit limit was setted and a information message was popped up.
Conclusion:	The test is successful.

Table 6: Test Table to set credit limit for Credit Card in BankCardGUI



Bank Card

ABC BANK

Thank You For Choosing Us

DebitCard Details

Balance Amount:	
Card ID:	
Bank Account:	
Issuer Bank:	
Client Name:	
PIN Number:	

CreditCard Details

Balance Amount:	120000
Card ID:	101
Bank Account:	SFBADV4AD23A3
Issuer Bank:	ABC BANK
Client Name:	Nick
CVC Number:	123
Interest Rate:	12
Expiration Date:	1 Jan 2023

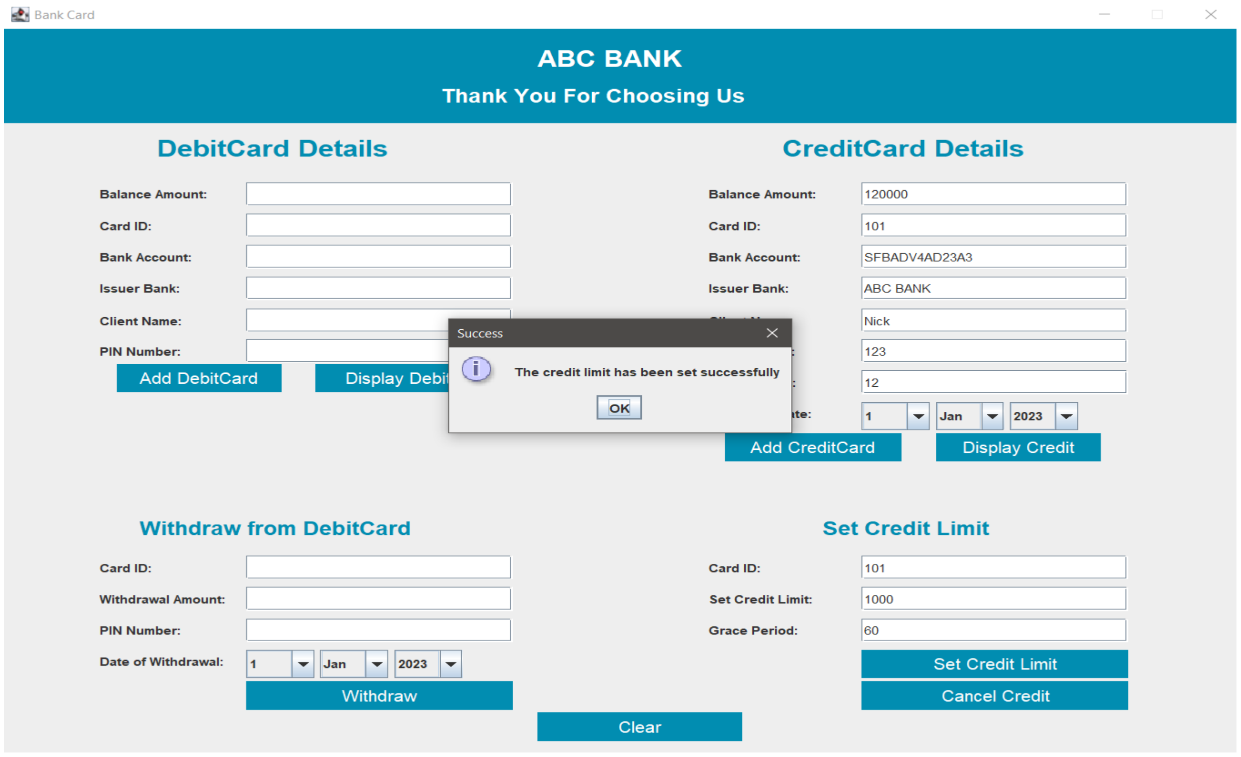
Withdraw from DebitCard

Card ID:	
Withdrawal Amount:	
PIN Number:	
Date of Withdrawal:	1 Jan 2023

Set Credit Limit

Card ID:	101
Set Credit Limit:	1000
Grace Period:	60

Figure 17: Setting The Credit Limit



Bank Card

ABC BANK

Thank You For Choosing Us

DebitCard Details

CreditCard Details

Balance Amount:	120000
Card ID:	101
Bank Account:	SFBADV4AD23A3
Issuer Bank:	ABC BANK
	Nick
	123
	12
	1 Jan 2023

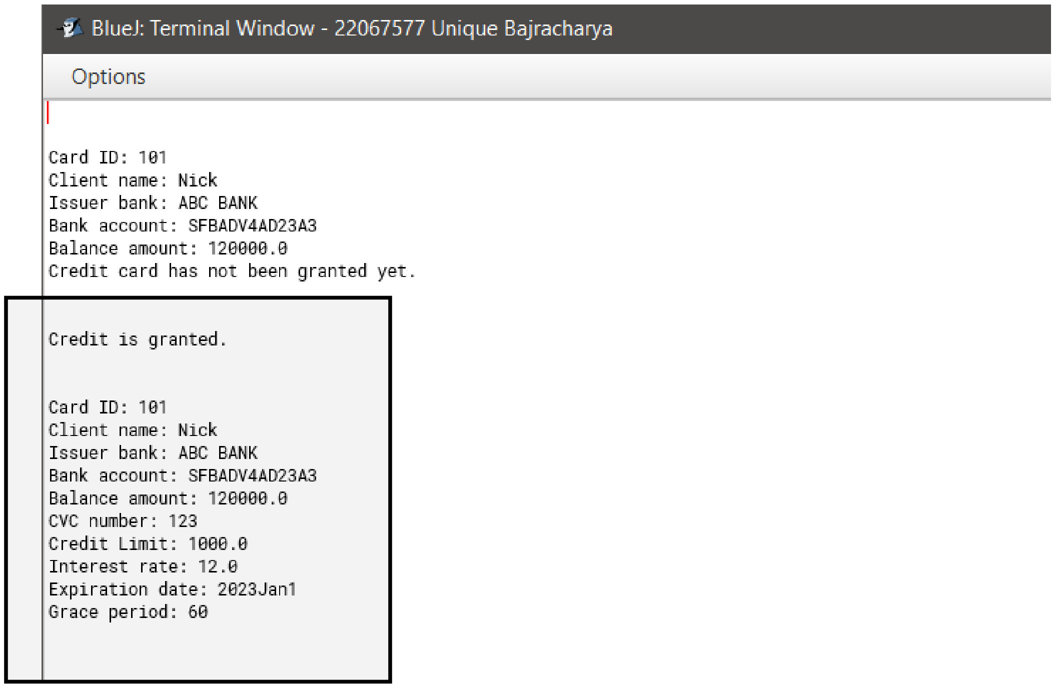
Success

The credit limit has been set successfully

Set Credit Limit

Card ID:	101
Set Credit Limit:	1000
Grace Period:	60

Figure 18: Success Message for setting Credit Limit



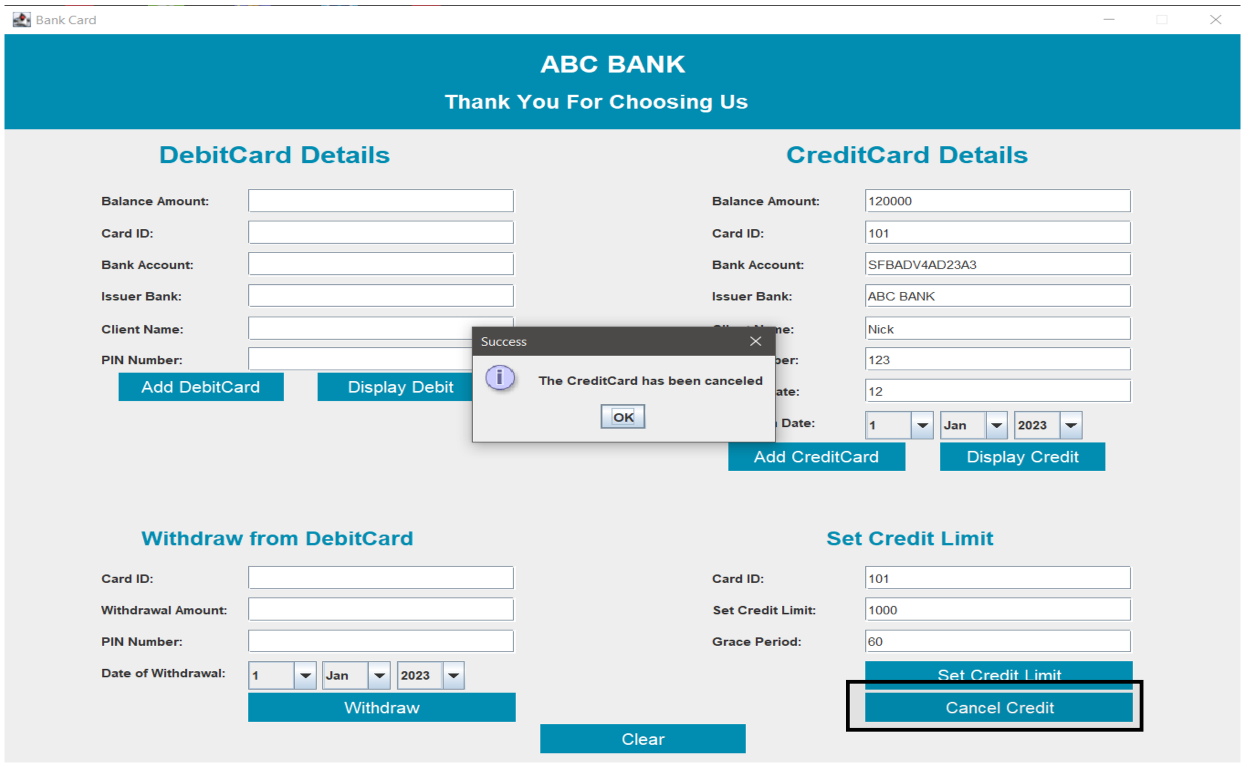
```
Card ID: 101
Client name: Nick
Issuer bank: ABC BANK
Bank account: SFBADV4AD23A3
Balance amount: 120000.0
Credit card has not been granted yet.

Credit is granted.


Card ID: 101
Client name: Nick
Issuer bank: ABC BANK
Bank account: SFBADV4AD23A3
Balance amount: 120000.0
CVC number: 123
Credit Limit: 1000.0
Interest rate: 12.0
Expiration date: 2023Jan1
Grace period: 60
```

Figure 19: Details after setting Credit Limit

f. Test - f

Test No.:	f
Objective:	Cancel CreditCard for CreditCard
Action:	• "Cancel Credit" button is pressed
Expected Result:	The CreditCard should be cancelled and a information message should pop up.
Actual Result:	The CreditCard was cancelled and a information message was popped up.
Conclusion:	The test is successful.

Table 7: Test Table to cancel Credit Card in BankCardGUI



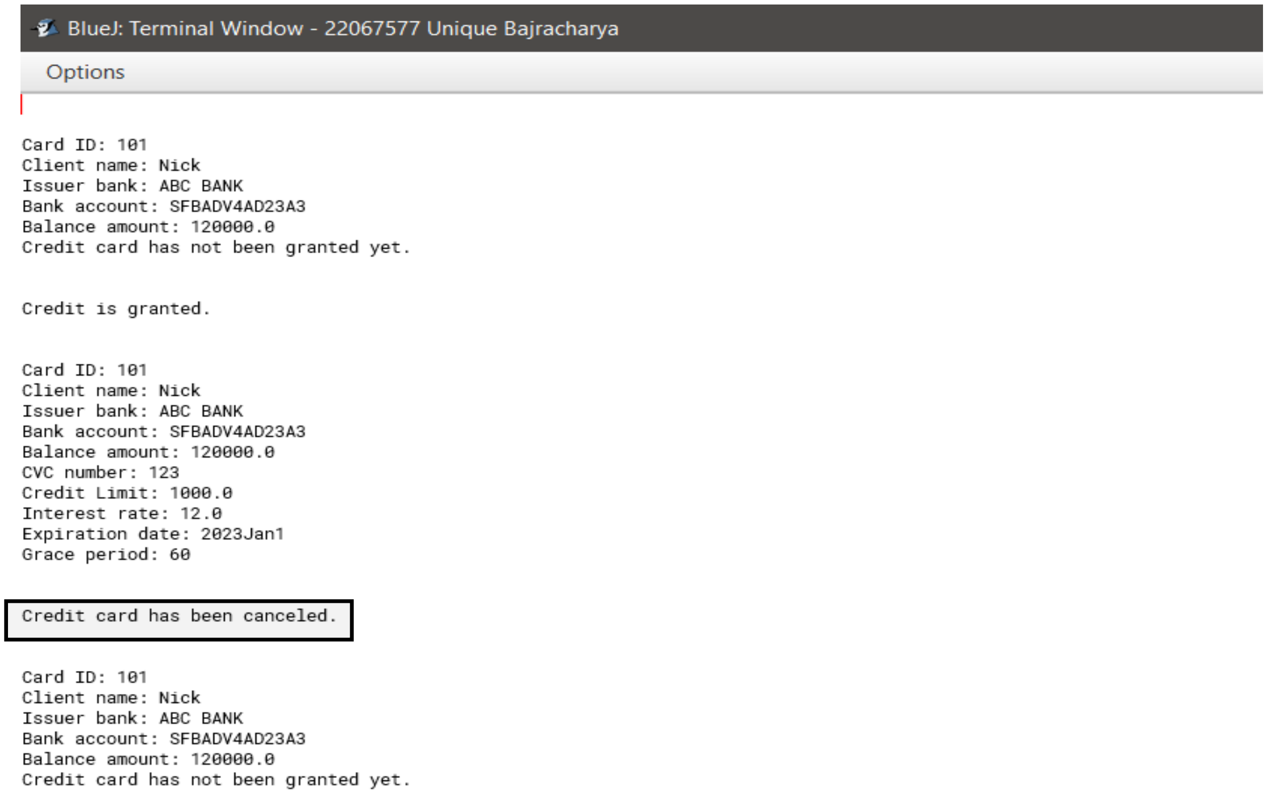
Balance Amount:	120000
Card ID:	101
Bank Account:	SFBADV4AD23A3
Issuer Bank:	ABC BANK
Client Name:	Nick
CVC Number:	123
Interest Rate:	12
Expiration Date:	1 Jan 2023

Card ID:	101
Set Credit Limit:	1000
Grace Period:	60

Card ID:	
Withdrawal Amount:	
PIN Number:	
Date of Withdrawal:	1 Jan 2023

The CreditCard has been canceled

Figure 20: Success message for Cancel Credit Card



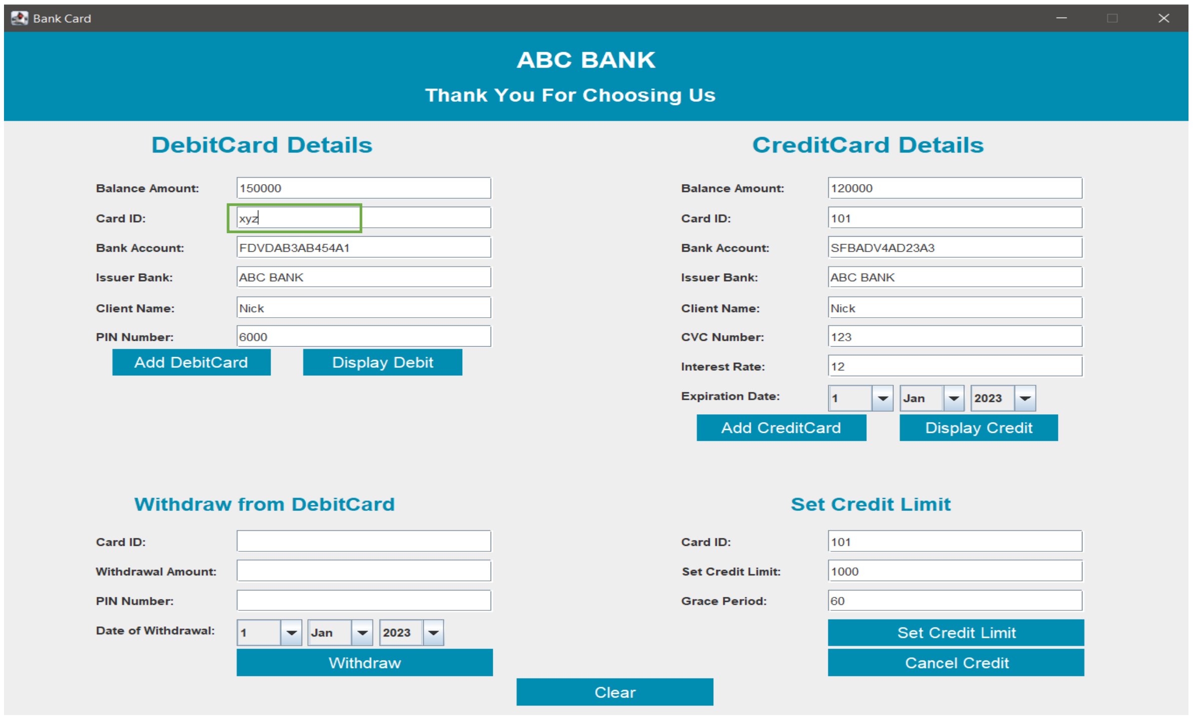
```
Card ID: 101
Client name: Nick
Issuer bank: ABC BANK
Bank account: SFBADV4AD23A3
Balance amount: 120000.0
Credit card has not been granted yet.


Credit is granted.


Card ID: 101
Client name: Nick
Issuer bank: ABC BANK
Bank account: SFBADV4AD23A3
Balance amount: 120000.0
CVC number: 123
Credit Limit: 1000.0
Interest rate: 12.0
Expiration date: 2023Jan1
Grace period: 60

Credit card has been canceled.

Card ID: 101
Client name: Nick
Issuer bank: ABC BANK
Bank account: SFBADV4AD23A3
Balance amount: 120000.0
Credit card has not been granted yet.
```

Figure 21: Details after Cancelling the Credit Card

3. Test - 3

Test No.:	3
Objective:	Trying to add DebitCard with String in card id
Action:	• Running add DebitCard button in GUI • Add DebitCard is run using the following values: Balance amount = 150000 Card id = xyz Bank Account = FDVDAB3AB454A1 Issuer Bank = ABC BANK Client Name = Nick PIN Number = 6000 • “Add DebitCard” button is pressed.
Expected Result:	A message should be displayed saying the format cannot be accepted.
Actual Result:	A message is displayed saying the format cannot be accepted.
Conclusion:	The test is successful.

Table 8: Test Table to add Debit Card with String in card ID



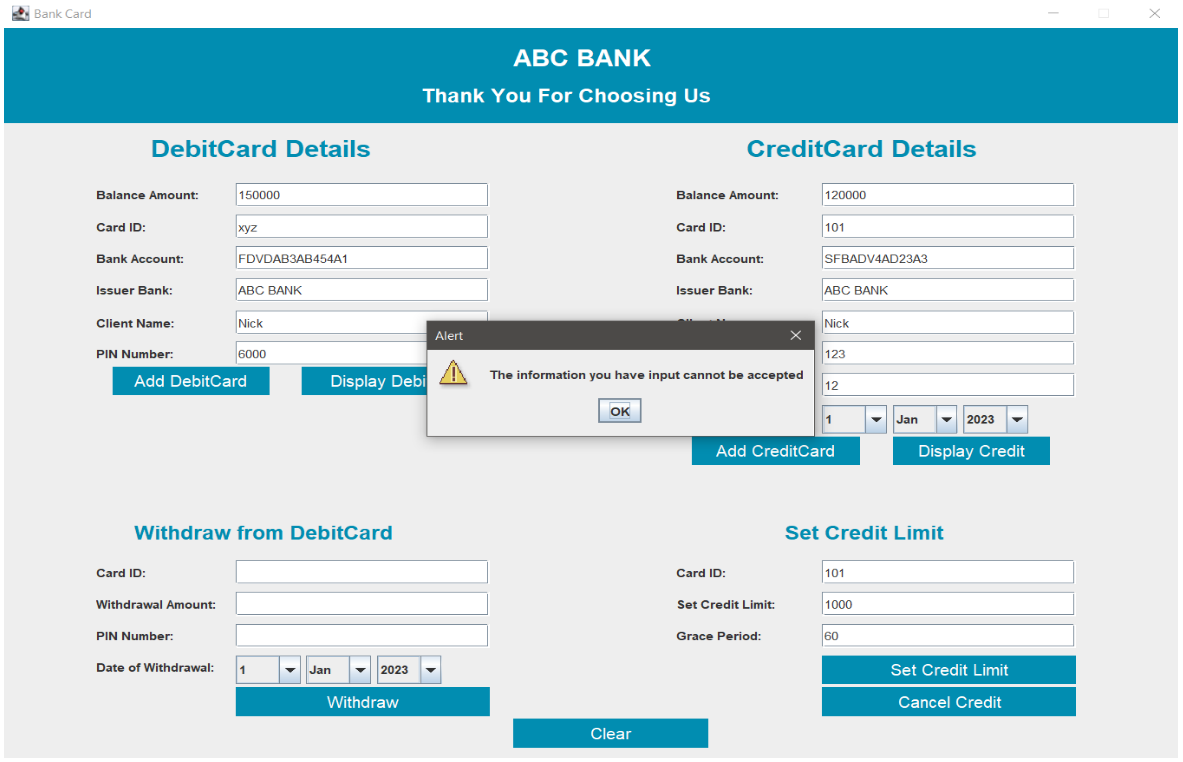
Balance Amount:	150000
Card ID:	xyz
Bank Account:	FDVDAB3AB454A1
Issuer Bank:	ABC BANK
Client Name:	Nick
PIN Number:	6000

Balance Amount:	120000
Card ID:	101
Bank Account:	SFBADV4AD23A3
Issuer Bank:	ABC BANK
Client Name:	Nick
CVC Number:	123
Interest Rate:	12
Expiration Date:	1 Jan 2023

Card ID:	
Withdrawal Amount:	
PIN Number:	
Date of Withdrawal:	1 Jan 2023

Card ID:	101
Set Credit Limit:	1000
Grace Period:	60

Figure 22: Entering Card ID as String Value



The information you have input cannot be accepted

Balance Amount:	150000
Card ID:	xyz
Bank Account:	FDVDAB3AB454A1
Issuer Bank:	ABC BANK
Client Name:	Nick
PIN Number:	6000

Balance Amount:	120000
Card ID:	101
Bank Account:	SFBADV4AD23A3
Issuer Bank:	ABC BANK
	Nick
	123
	12
	1 Jan 2023

Card ID:	
Withdrawal Amount:	
PIN Number:	
Date of Withdrawal:	1 Jan 2023

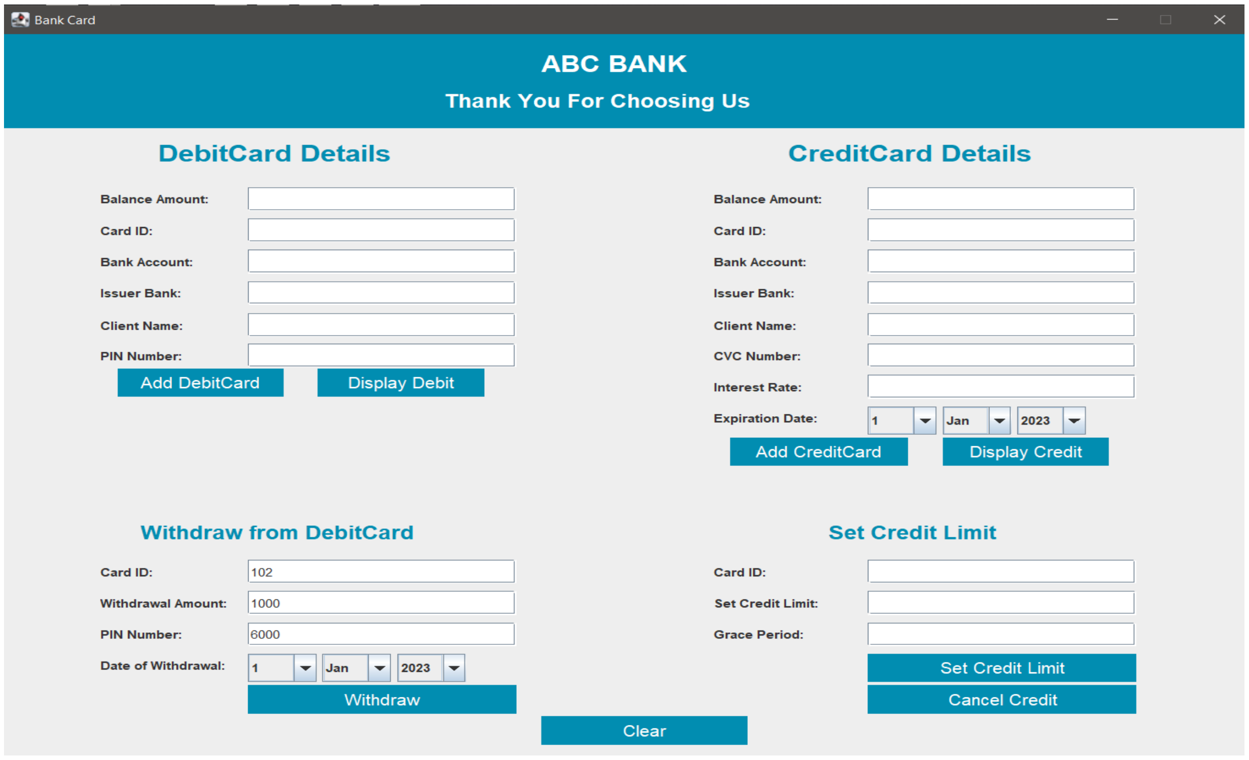
Card ID:	101
Set Credit Limit:	1000
Grace Period:	60

Figure 23: Error message showing unacceptable input

4. Test - 4

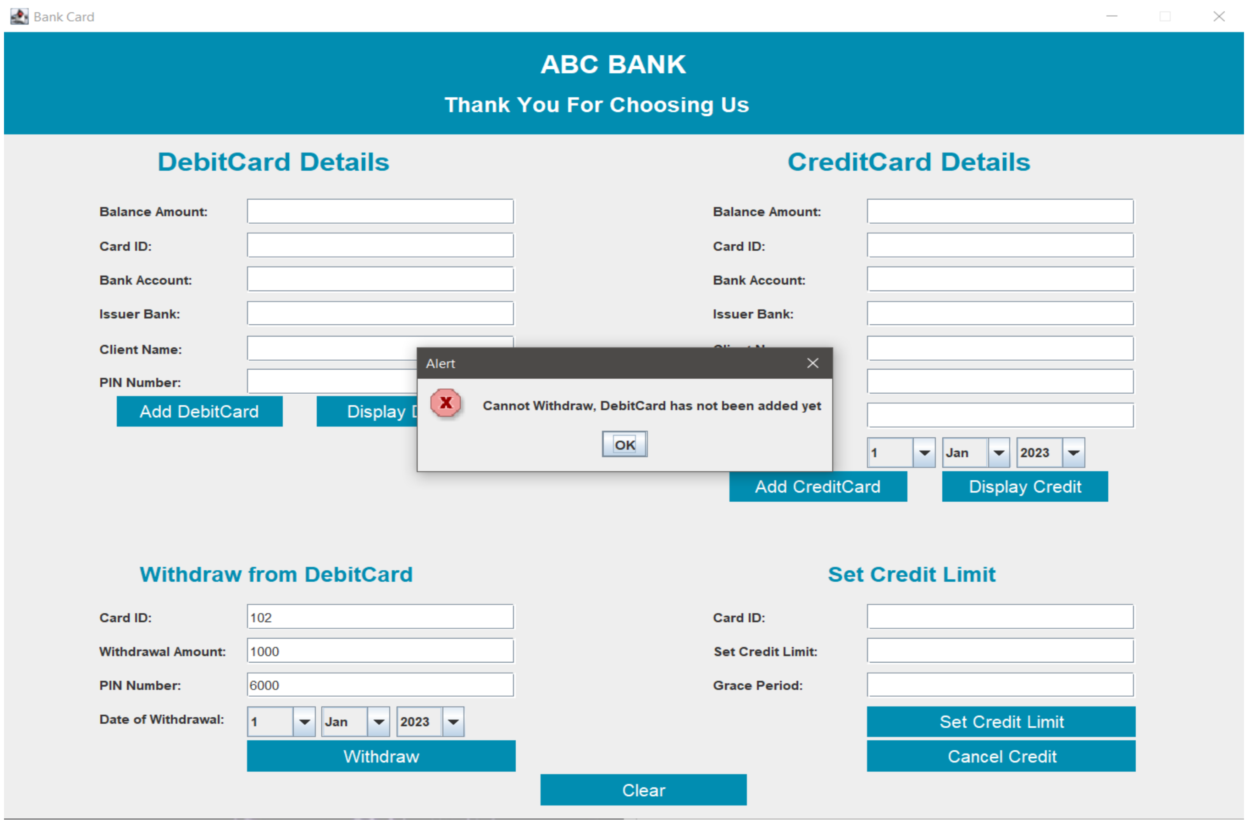
Test No.:	4
Objective:	Trying to withdraw from DebitCard without adding the DebitCard first.
Action:	• Running withdraw button in GUI. • The DebitCard details are kept empty. • The withdraw button is run using the following values: Card ID = 102 Withdrawal Amount = 1000 PIN Number = 6000 Date of Withdrawal = 1st jan 2023.
Expected Result:	A message should be displayed saying the DebitCard has not been added yet.
Actual Result:	A message was displayed saying the debit card has not been displayed yet.
Conclusion:	The test is successful.

Table 9: Test Table to withdraw amount from Debit Card in BankCardGUI without adding the DebitCard first



Card ID:	102
Withdrawal Amount:	1000
PIN Number:	6000
Date of Withdrawal:	1 Jan 2023

Figure 24: Withdrawing without adding DebitCard Details



Cannot Withdraw, DebitCard has not been added yet

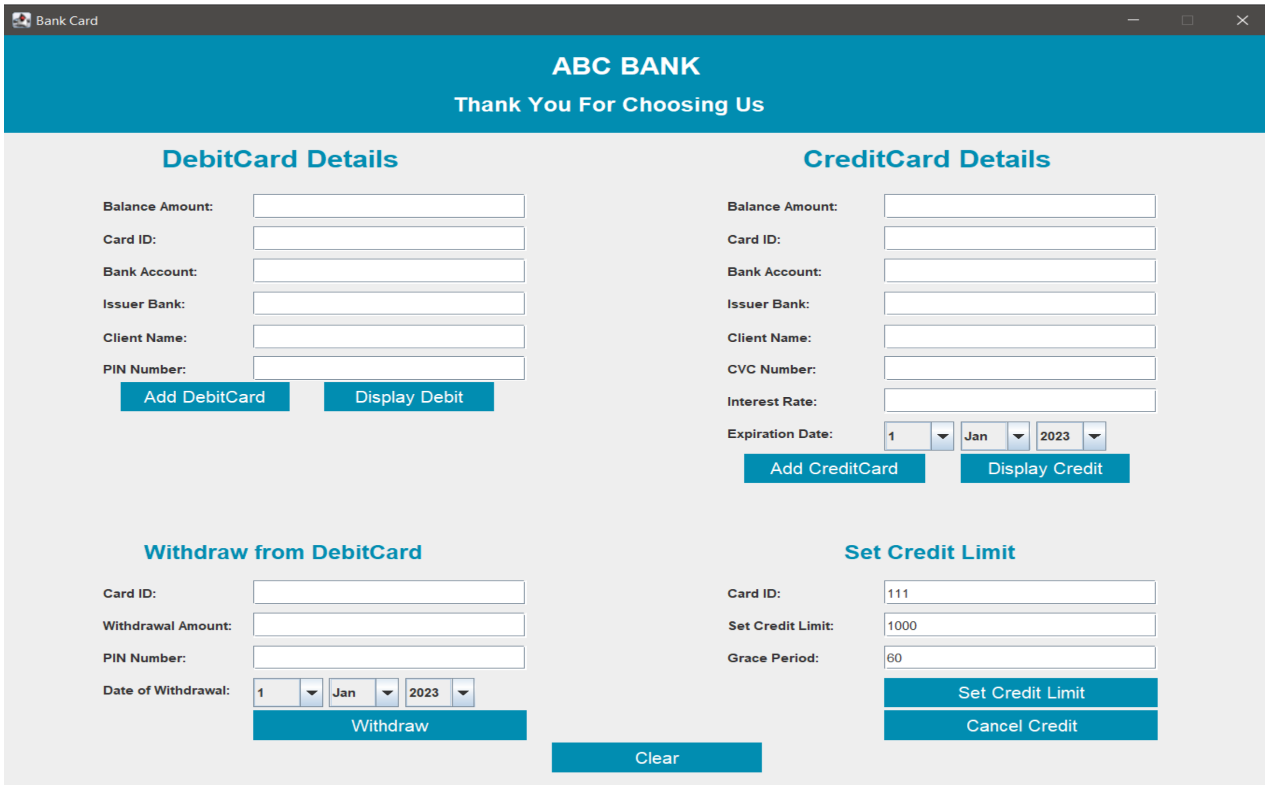
Card ID:	102
Withdrawal Amount:	1000
PIN Number:	6000
Date of Withdrawal:	1 Jan 2023

Figure 25: Error message showing Debit Card has not been added yet

5. Test – 5

Test No.:	5
Objective:	Trying to set credit limit without adding Credit card first.
Action:	• Running setCreditLimit button in GUI. • The CreditCard details are kept empty. • The withdraw button is run using the following values: Card ID = 111 Set Credit Limit = 1000 Grace Period = 60 • "Set Credit Limit" button is pressed.
Expected Result:	A message should be displayed saying the Credit card has not been added yet.
Actual Result:	A message was displayed saying the Credit card has not been displayed yet.
Conclusion:	The test is successful.

Table 10: Test Table to set credit limit in BankCardGUI without adding Credit Card first



Bank Card

ABC BANK

Thank You For Choosing Us

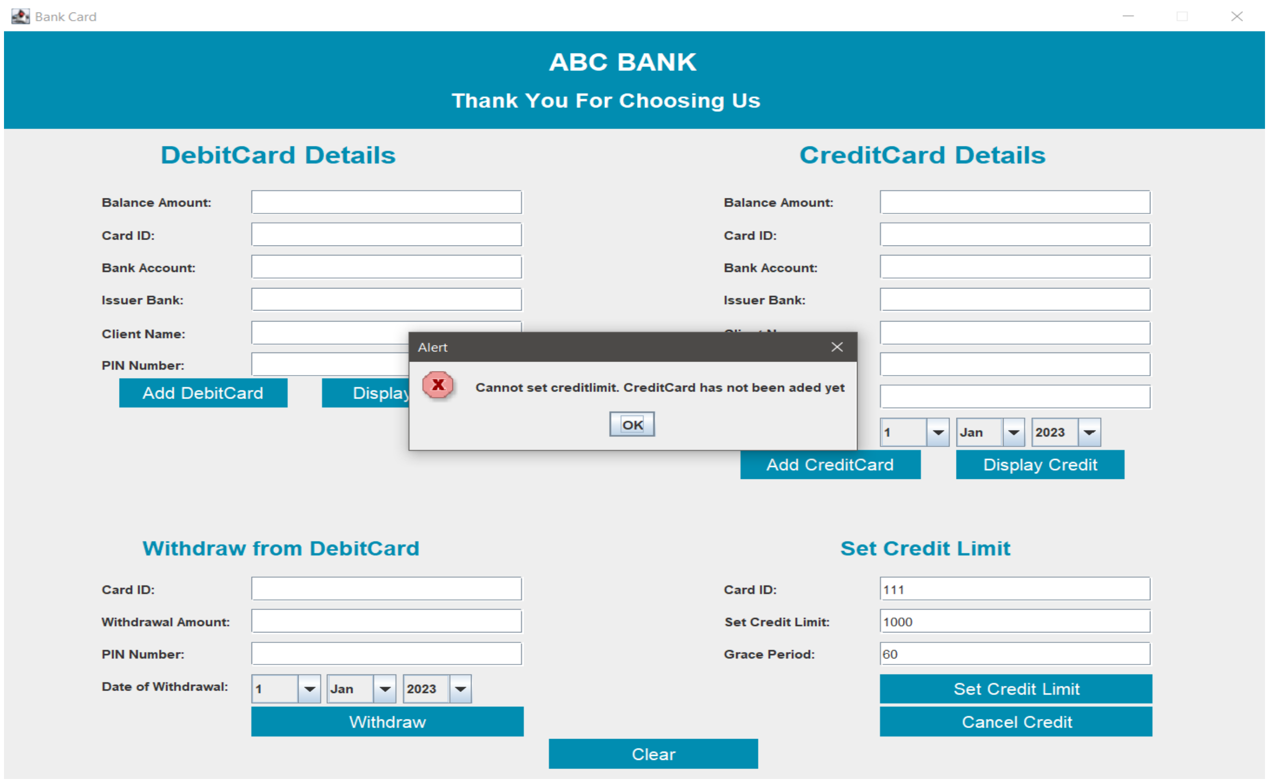
DebitCard Details	
Balance Amount:	
Card ID:	
Bank Account:	
Issuer Bank:	
Client Name:	
PIN Number:	

CreditCard Details	
Balance Amount:	
Card ID:	
Bank Account:	
Issuer Bank:	
Client Name:	
CVC Number:	
Interest Rate:	
Expiration Date:	1 Jan 2023

Withdraw from DebitCard	
Card ID:	
Withdrawal Amount:	
PIN Number:	
Date of Withdrawal:	1 Jan 2023

Set Credit Limit	
Card ID:	111
Set Credit Limit:	1000
Grace Period:	60

Figure 26: Setting Credit Limit without adding CreditCard Details



Bank Card

ABC BANK

Thank You For Choosing Us

Set Credit Limit	
Card ID:	111
Set Credit Limit:	1000
Grace Period:	60

Alert

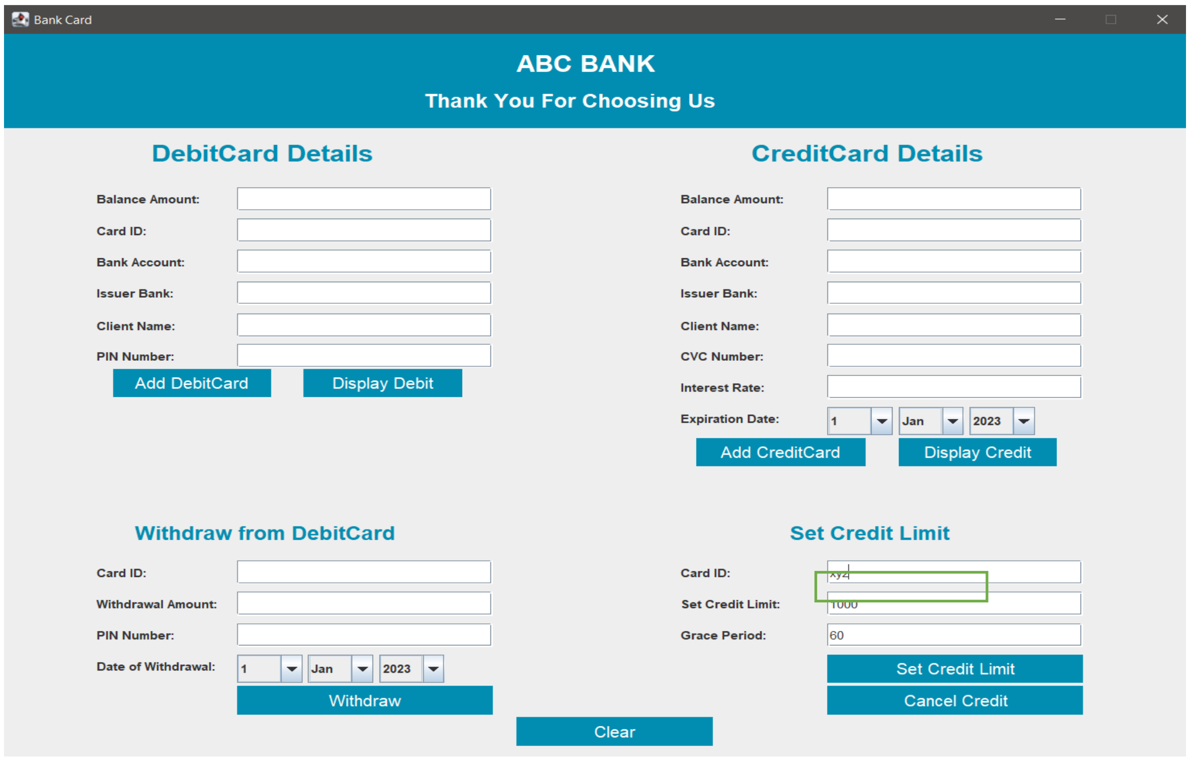
Cannot set creditlimit. CreditCard has not been aded yet

Figure 27: Error message showing Debit Card has not been added yet

6. Test – 6

Test No.:	6
Objective:	Trying to set credit limit with String in card id
Action:	• Running Set Credit Limit button in GUI • setCreditCard is run using the following values: Card id = xyz Set Credit Limit = 1000 Grace Period = 60 • “Set Credit Limit” button is pressed.
Expected Result:	A message should be displayed saying the format cannot be accepted.
Actual Result:	A message is displayed saying the format cannot be accepted.
Conclusion:	The test is successful.

Table 11: Test Table to set set credit limit with String in Card ID

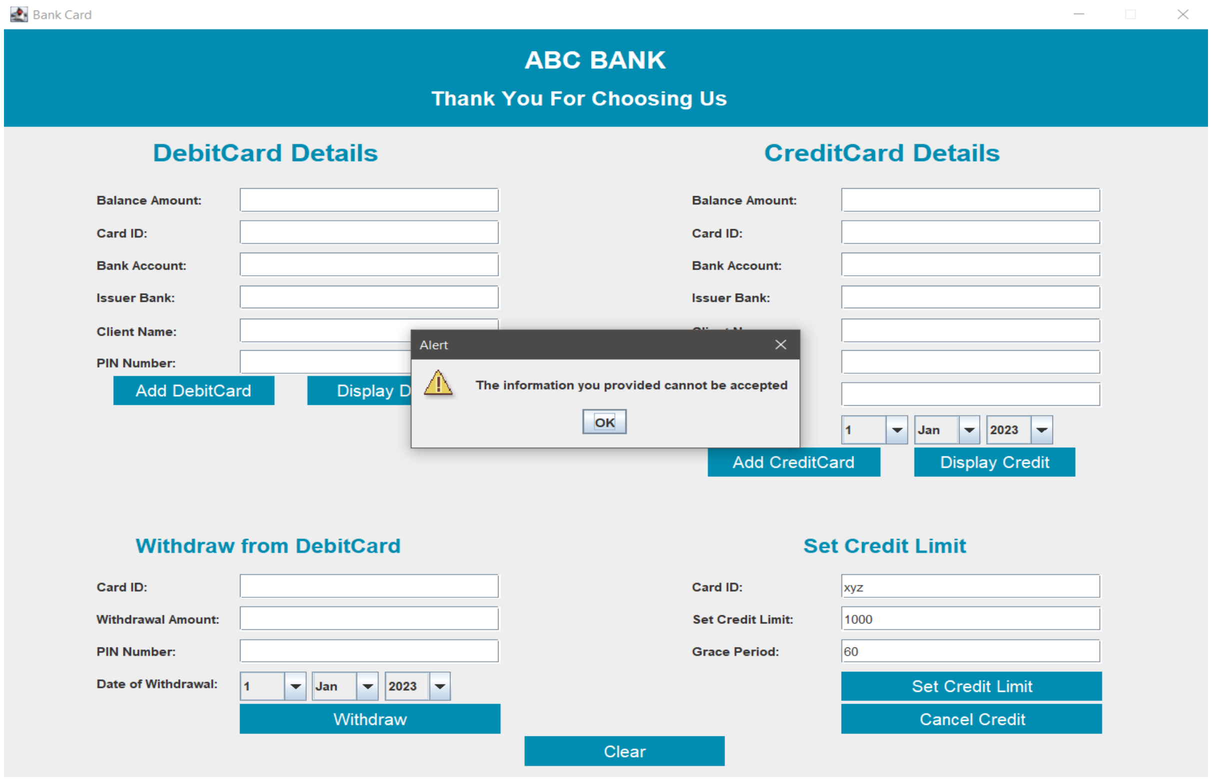


Card ID: xyz

Set Credit Limit: 1000

Grace Period: 60

Figure 28: Entering Card ID as String Value



The information you provided cannot be accepted

Card ID: xyz

Set Credit Limit: 1000

Grace Period: 60

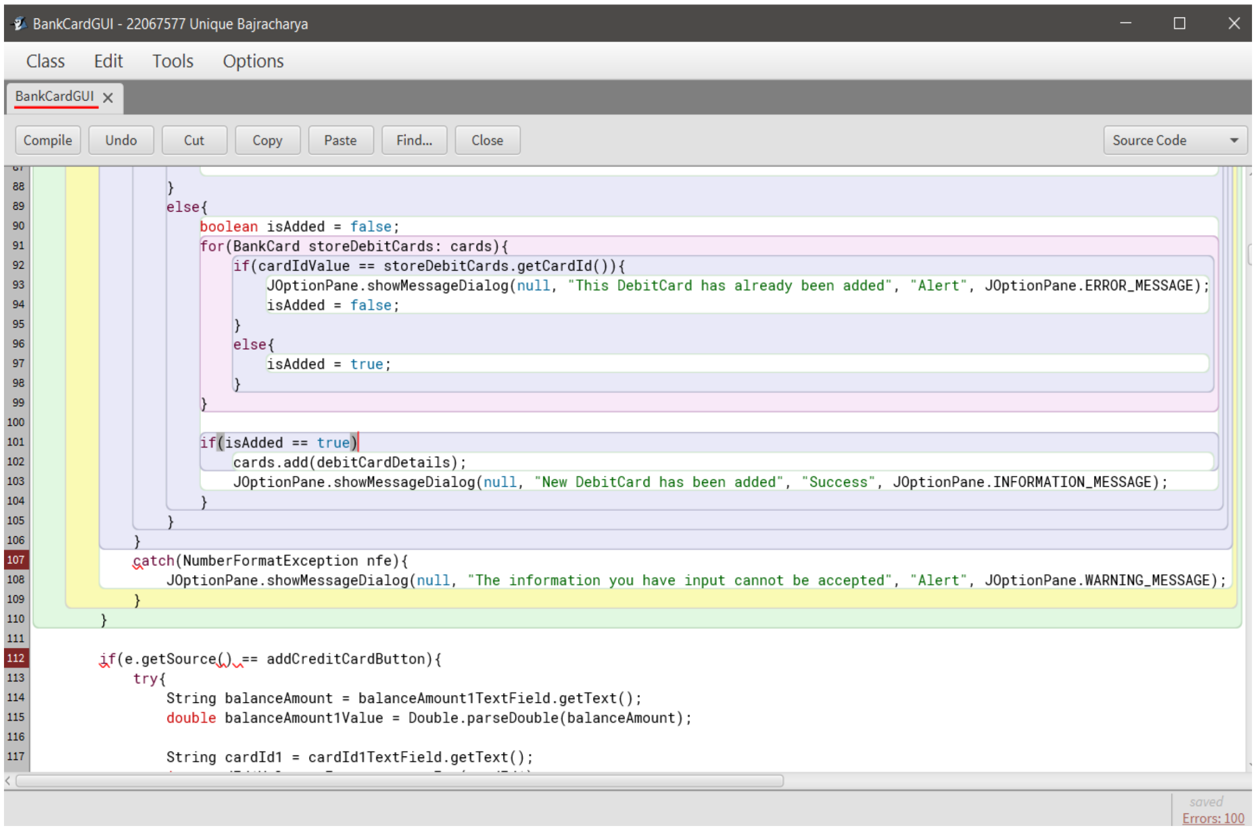
Figure 29: Error message showing unacceptable input

6. Error Detection and Correction

1. Syntax Error

A syntax error occurs in Java when there is an error in the source code of a program. Syntax errors, like grammatical errors in language, occur when the program's syntax rules are broken. An extra semicolon at the end of a function, for example, could result in a syntactical problem. A syntax problem can prevent a program from compiling or running properly, resulting in unpredictable behavior or even program crashes. As a result, it is critical to identify and correct syntax problems as soon as possible during the development process. (George, 2022)

Detection of Syntax Error

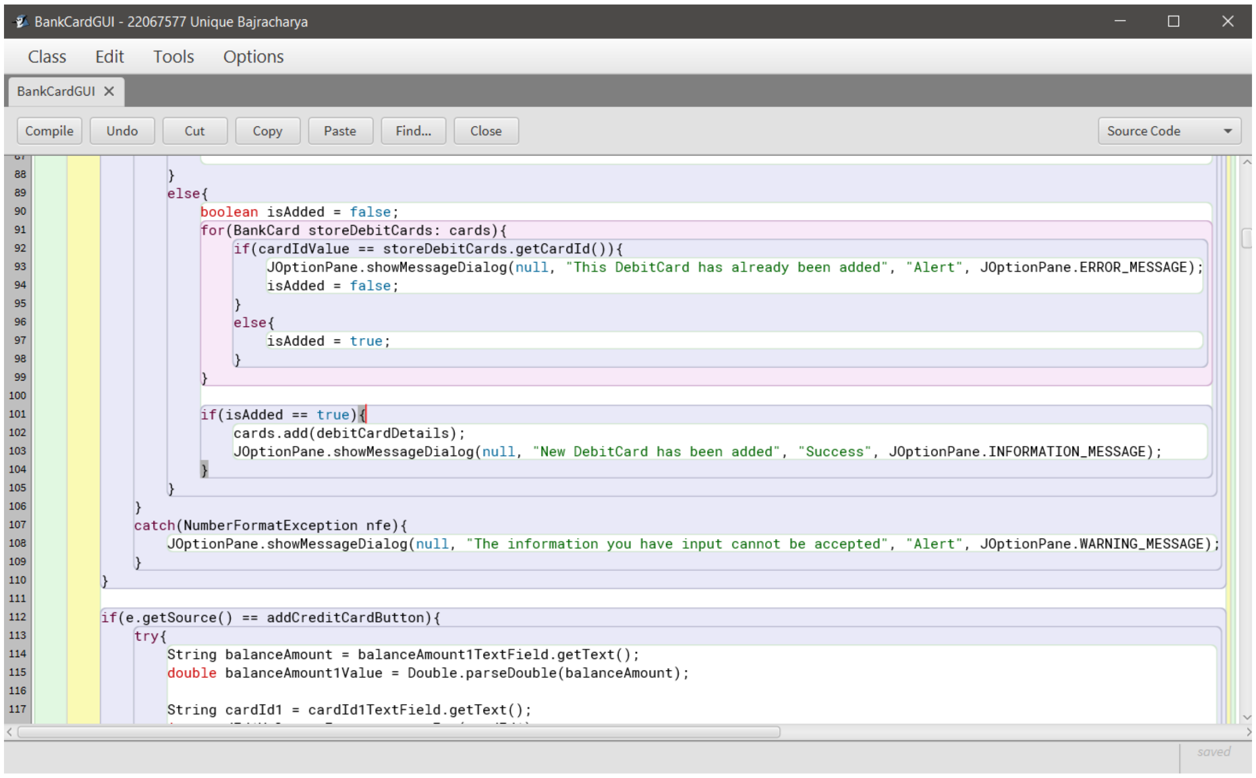


```
        }
        else{
            boolean isAdded = false;
            for(BankCard storeDebitCards: cards){
                if(cardIdValue == storeDebitCards.getCardId()){
                    JOptionPane.showMessageDialog(null, "This DebitCard has already been added", "Alert", JOptionPane.ERROR_MESSAGE);
                    isAdded = false;
                }
                else{
                    isAdded = true;
                }
            }

            if(isAdded == true)
                cards.add(debitCardDetails);
                JOptionPane.showMessageDialog(null, "New DebitCard has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
    catch(NumberFormatException nfe){
        JOptionPane.showMessageDialog(null, "The information you have input cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);
    }
}

if(e.getSource() == addCreditCardButton){
    try{
        String balanceAmount = balanceAmount1TextField.getText();
        double balanceAmount1Value = Double.parseDouble(balanceAmount);

        String cardId1 = cardId1TextField.getText();
```

Figure 30: Detection of Syntax Error

Figure 28 shows that bluej discovered an error in the program when compiling it. This is referred to as a syntactical error. The mistake in this code is that the final curly brace for the else statement is misplaced, resulting in an improperly nested if expression. These types of errors can easily be corrected with the proper coding syntax.

Correction of Syntax Error



```java
        }
        else{
            boolean isAdded = false;
            for(BankCard storeDebitCards: cards){
                if(cardIdValue == storeDebitCards.getCardId()){
                    JOptionPane.showMessageDialog(null, "This DebitCard has already been added", "Alert", JOptionPane.ERROR_MESSAGE);
                    isAdded = false;
                }
                else{
                    isAdded = true;
                }
            }

            if(isAdded == true){
                cards.add(debitCardDetails);
                JOptionPane.showMessageDialog(null, "New DebitCard has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
    catch(NumberFormatException nfe){
        JOptionPane.showMessageDialog(null, "The information you have input cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);
    }
}

if(e.getSource() == addCreditCardButton){
    try{
        String balanceAmount = balanceAmount1TextField.getText();
        double balanceAmount1Value = Double.parseDouble(balanceAmount);

        String cardId1 = cardId1TextField.getText();
```

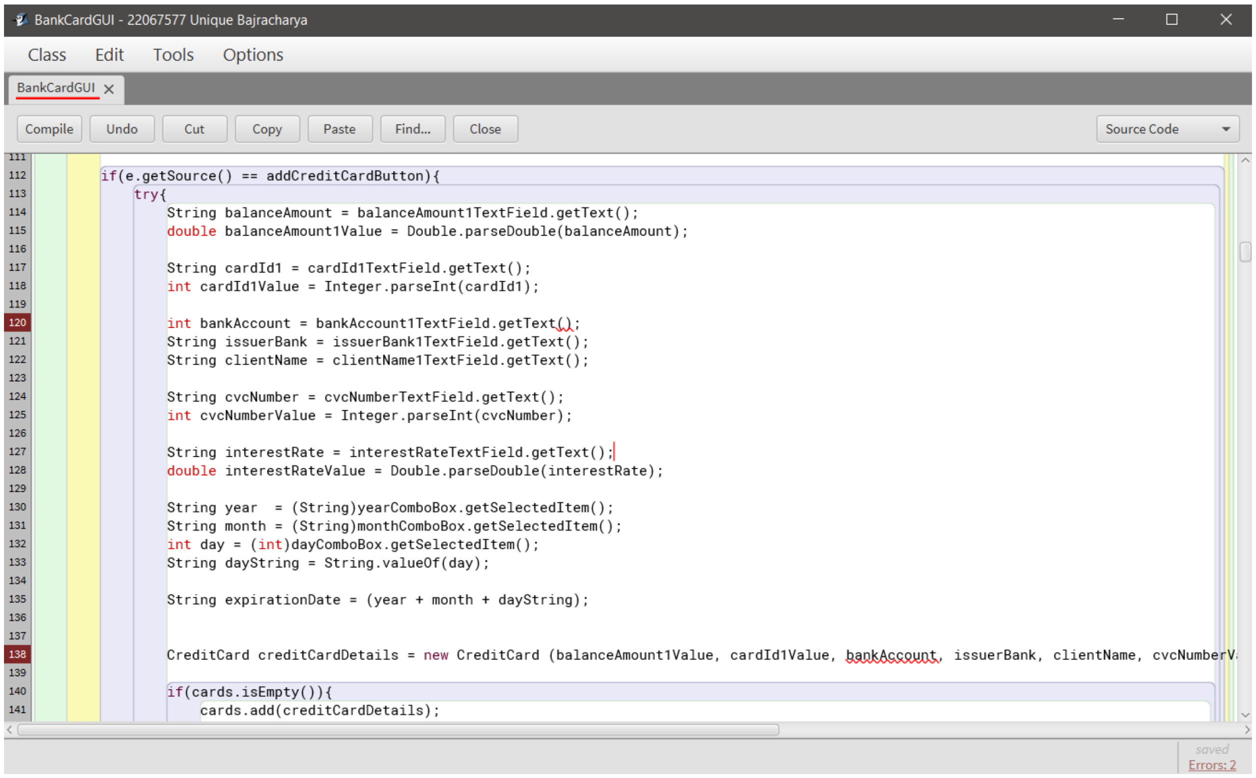
Figure 31: Correction of Syntax Error

The problem that bluej displayed in figure 29 has been fixed. The curly brace is a crucial component of the Java syntax that aids the compiler in understanding the code's structure. Therefore, by including a curly brace behind the if statement, the mistake has been fixed.

2. Semantic Error

It is often observed that many people find it difficult to understand the difference between syntactical and semantic problems in Java code. Syntax errors are problems in the code's structure, but semantic errors are mistakes in how the code is used despite having a correct structure. These mistakes are frequently harder to spot than syntactic mistakes. (Mueller, Semantic Errors in Java, 2016)

Detection of Semantic Error

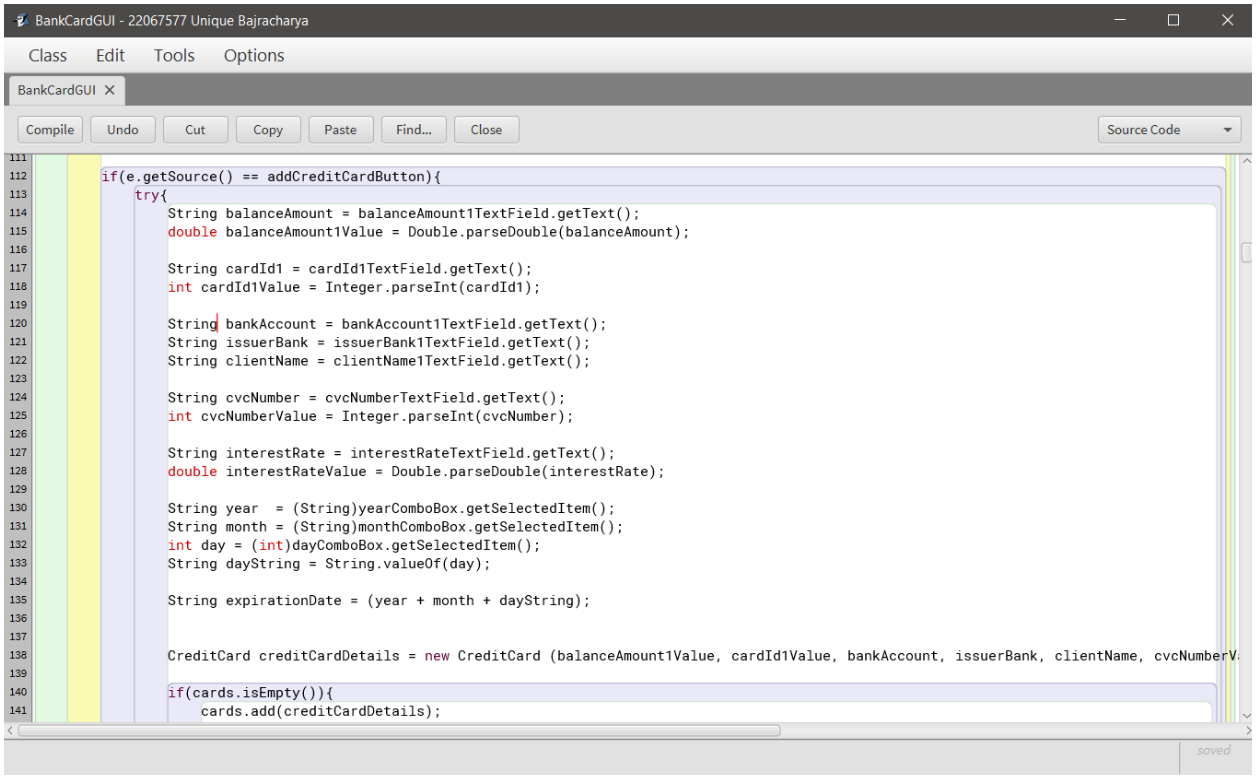

BankCardGUI - 22067577 Unique Bajracharya
Class Edit Tools Options
BankCardGUI
Compile Undo Cut Copy Paste Find... Close
Source Code
saved
Errors: 2

```
if(e.getSource() == addCreditCardButton){
    try{
        String balanceAmount = balanceAmount1TextField.getText();
        double balanceAmount1Value = Double.parseDouble(balanceAmount);

        String cardId1 = cardId1TextField.getText();
        int cardId1Value = Integer.parseInt(cardId1);

        int bankAccount = bankAccount1TextField.getText();
        String issuerBank = issuerBank1TextField.getText();
        String clientName = clientName1TextField.getText();

        String cvcNumber = cvcNumberTextField.getText();
        int cvcNumberValue = Integer.parseInt(cvcNumber);

        String interestRate = interestRateTextField.getText();
        double interestRateValue = Double.parseDouble(interestRate);

        String year  = (String)yearComboBox.getSelectedItem();
        String month = (String)monthComboBox.getSelectedItem();
        int day = (int)dayComboBox.getSelectedItem();
        String dayString = String.valueOf(day);

        String expirationDate = (year + month + dayString);


        CreditCard creditCardDetails = new CreditCard (balanceAmount1Value, cardId1Value, bankAccount, issuerBank, clientName, cvcNumberV

        if(cards.isEmpty()){
            cards.add(creditCardDetails);
```

Figure 32: Detection of Semantic Error

While constructing the software, bluej found the issue shown in figure 30. Semantic errors include mistakes like this. While the grammar in these errors is accurate, the code isn't being used in the right way. By comprehending the error message that bluej displays, they are readily rectified.

Correction of Semantic Error


BankCardGUI - 22067577 Unique Bajracharya
Class Edit Tools Options
BankCardGUI
Compile Undo Cut Copy Paste Find... Close
Source Code
saved

```
if(e.getSource() == addCreditCardButton){
    try{
        String balanceAmount = balanceAmount1TextField.getText();
        double balanceAmount1Value = Double.parseDouble(balanceAmount);

        String cardId1 = cardId1TextField.getText();
        int cardId1Value = Integer.parseInt(cardId1);

        String bankAccount = bankAccount1TextField.getText();
        String issuerBank = issuerBank1TextField.getText();
        String clientName = clientName1TextField.getText();

        String cvcNumber = cvcNumberTextField.getText();
        int cvcNumberValue = Integer.parseInt(cvcNumber);

        String interestRate = interestRateTextField.getText();
        double interestRateValue = Double.parseDouble(interestRate);

        String year  = (String)yearComboBox.getSelectedItem();
        String month = (String)monthComboBox.getSelectedItem();
        int day = (int)dayComboBox.getSelectedItem();
        String dayString = String.valueOf(day);

        String expirationDate = (year + month + dayString);


        CreditCard creditCardDetails = new CreditCard (balanceAmount1Value, cardId1Value, bankAccount, issuerBank, clientName, cvcNumberV

        if(cards.isEmpty()){
            cards.add(creditCardDetails);
```

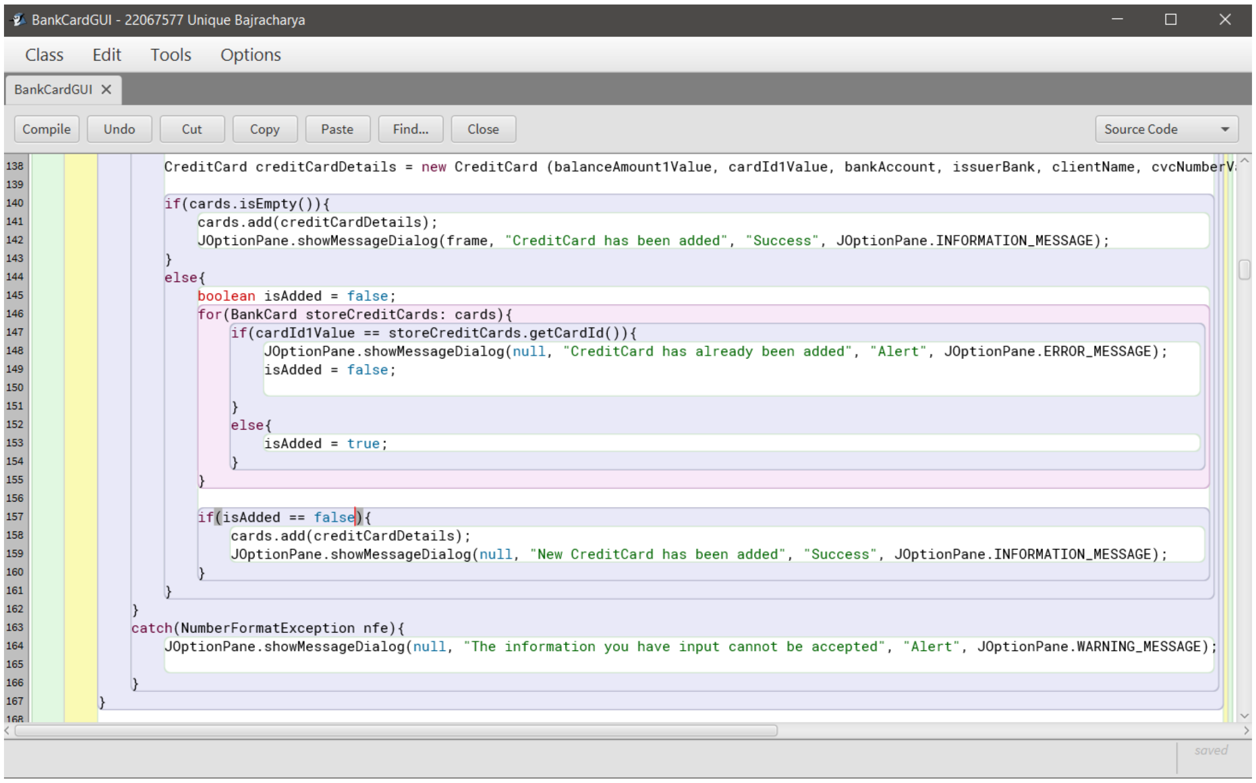
Figure 33: Correction of Semantic Error

The problem that bluej displayed in figure 31 has been fixed. "int" is transformed to "String" as a data type. This mistake was discovered by Bluej because the bank account's value was first initialized in the super class as a String and could not be changed to a "int". The mistake was fixed by changing to String, and after that, the program was compiled.

3. Logical Error

Java programming can sometimes have logical errors that are hard to identify, as they don’t involve any sort of coding errors in the language syntax. logical errors in Java programming occasionally occur and can be challenging to spot. The application might operate without any problems, but it might not carry out the task as intended. As a result, it can be difficult to find these problems, and the compiler does not catch them. Programs may experience unexpected behavior because of logical errors that occur due to faults in the logic of the program. (Mueller, Logical Errors in Java, 2016)

Detection of Logical Error



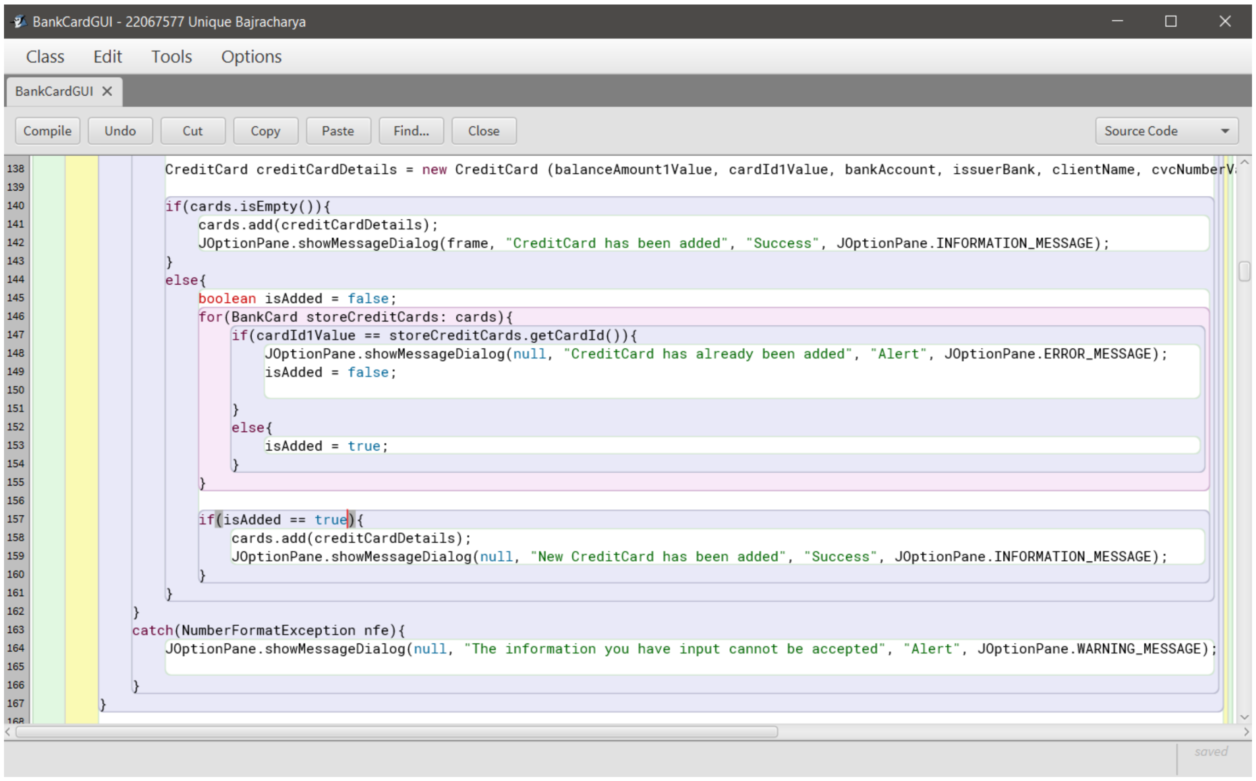
```
        CreditCard creditCardDetails = new CreditCard (balanceAmount1Value, cardId1Value, bankAccount, issuerBank, clientName, cvcNumberV

        if(cards.isEmpty()){
            cards.add(creditCardDetails);
            JOptionPane.showMessageDialog(frame, "CreditCard has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
        }
        else{
            boolean isAdded = false;
            for(BankCard storeCreditCards: cards){
                if(cardId1Value == storeCreditCards.getCardId()){
                    JOptionPane.showMessageDialog(null, "CreditCard has already been added", "Alert", JOptionPane.ERROR_MESSAGE);
                    isAdded = false;

                }
                else{
                    isAdded = true;
                }
            }

            if(isAdded == false){
                cards.add(creditCardDetails);
                JOptionPane.showMessageDialog(null, "New CreditCard has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
    catch(NumberFormatException nfe){
        JOptionPane.showMessageDialog(null, "The information you have input cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);

    }
}
```

BankCardGUI - 22067577 Unique Bajracharya
Class Edit Tools Options
BankCardGUI
Compile Undo Cut Copy Paste Find... Close
Source Code
saved

Figure 34: Detection of Logical Error

Bluej found a programming issue in figure 32 during compilation of the program. Logical error is the name given to this kind of error. The program's logical mistake prevented the intended output from being produced during execution, despite the fact that the program's syntax is correct and the bluej compiles it. No problem was identified by bluej and the program was run even when false was written in the above code in place of true. But the desired result was not achieved.

Correction of Logical Error


BankCardGUI - 22067577 Unique Bajracharya
Class Edit Tools Options
BankCardGUI
Compile Undo Cut Copy Paste Find... Close
Source Code
saved

```
        CreditCard creditCardDetails = new CreditCard (balanceAmount1Value, cardId1Value, bankAccount, issuerBank, clientName, cvcNumberV

        if(cards.isEmpty()){
            cards.add(creditCardDetails);
            JOptionPane.showMessageDialog(frame, "CreditCard has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
        }
        else{
            boolean isAdded = false;
            for(BankCard storeCreditCards: cards){
                if(cardId1Value == storeCreditCards.getCardId()){
                    JOptionPane.showMessageDialog(null, "CreditCard has already been added", "Alert", JOptionPane.ERROR_MESSAGE);
                    isAdded = false;

                }
                else{
                    isAdded = true;
                }
            }

            if(isAdded == true){
                cards.add(creditCardDetails);
                JOptionPane.showMessageDialog(null, "New CreditCard has been added", "Success", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
    catch(NumberFormatException nfe){
        JOptionPane.showMessageDialog(null, "The information you have input cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);

    }
}
```

Figure 35: Correction of Logical Error

In the figure 33, the error displayed by bluej has been corrected by writing true instead of false and program has been compiled. The desired output was achieved this time and the logical mistake was eliminated once the code was corrected.

7. Conclusion

I have learned a lot by working on this project. I now have a great deal of understanding of the challenges in the real world. I now know more about catch-block-catchable exceptions like NumberFormat and the GUI. Additionally, I now know more about object-oriented programming and inheritance. I've learned how to downcast an object using the keyword "instanceof" which return the Boolean value thanks to this project.

I had several obstacles when working on my project. Fortunately, my teacher was able to clarify how to evaluate the assignments, and my friends were crucial in assisting me in comprehending how to use the buttons effectively. My teachers were also always available in clarifying any misunderstanding that occurred throughout the project. Despite my best efforts, however the button functionality caused me some problems while testing the program. However, I was able to resolve these problems by conducting research online and applying the solutions that I found. Lastly, the course material on GUI functionality was new to me, and I found it somewhat challenging to grasp the specifics of how it operates. However. I remained committed to my project and preserved through the difficulties. By utilizing the available resources, I was able to overcome the obstacles and achieve success in my project.

The assignment is now over, and I now have the skills to write simple program code. The project taught me technical skills as well as the importance of patience and working under pressure. Overall, this assignment was an excellent learning opportunity that helped me to improve my basic coding skills and gather knowledge that would be useful in my future studies.

8. Bibliography

Airth, M. (2015). *https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html*. Retrieved January 19, 2023, from https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html

George, E. (2022). *What is a Syntax Error? How To Fix It*. Retrieved January 19, 2023, from https://clouddevelop.org/syntax-error/

Mueller, J. P. (2016). *Logical Errors in Java*. Retrieved January 19, 2023, from https://www.dummies.com/article/technology/programming-web-design/java/logical-errors-in-java-153712/

Mueller, J. P. (2016). *Semantic Errors in Java*. Retrieved January 19, 2023, from https://www.dummies.com/article/technology/programming-web-design/java/semantic-errors-in-java-153699/

Pedamkar, P. (2022). *Class Diagram*. Retrieved january 19, 2023, from https://www.educba.com/class-diagram/

9. Appendix

Code of the BankCardGUI class

```
/**
 * Write a description of class BankCardGUI here.
 *
 * @author (22067577 Unique Bajracharya)
 * @version (1.0.0)
 */

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

public class BankCardGUI implements ActionListener
{
   //declar all the componentes here
   //-----declaring components for debit card----//
   private JFrame frame;
   private JPanel panel;

   //----------------------Declare the components Debit card---------------------//
   private JLabel bankCardLabel, debitCardDetailsLabel, balanceAmountLabel,
cardIdLabel, bankAccountLabel, issuerBankLabel, clientNameLabel, pinNumberLabel,
debitCardLabel, cardId3Label, withdrawalAmountLabel, pinNumber1Label,
   dateOfWithdrawalLabel;
```

```
    private JTextField balanceAmountTextField, cardIdTextField, bankAccountTextField,
issuerBankTextField, clientNameTextField, pinNumberTextField, debitcardTextField,
withdrawalAmountTextField,

    pinNumber1TextField, cardId3TextField;

    private JButton addDebitCardButton, displayDebitCardButton, withdrawButton;

    private JComboBox debitDayComboBox, debitMonthComboBox,
debitYearComboBox;


    //-----------------------Declare the components of Credit card-------------------//

    private JLabel creditCardDetailsLabel, balanceAmount1Label, cardId1Label,
bankAccount1Label, issuerBank1Label, clientName1Label, cvcNumberLabel,
interestRateLabel, expirationDateLabel, setCreditLimitLabel,

    cardId4Label, creditCardLabel, gracePeriodLabel;

    private JTextField balanceAmount1TextField, issuerBank1TextField, cardId1TextField,
bankAccount1TextField, clientName1TextField, cvcNumberTextField,
interestRateTextField,

    cardId4TextField, setCreditLimitTextField, gracePeriodTextField;

    private JComboBox dayComboBox, monthComboBox, yearComboBox;

    private JButton addCreditCardButton, displayCreditCardButton, setCreditLimitButton,
cancelCreditButton, clearButton;


    //////Array list to store the details of the bank cards/////

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



    public BankCardGUI(){

        frame = new JFrame();


        panel = new JPanel();

        panel.setLayout(null);

        panel.setBackground(new Color(1, 141, 177)); // Set the background color
        panel.setBounds(0, 0, 1100, 100);

        frame.add(panel);

        bankCardLabel = new JLabel("ABC BANK");
        bankCardLabel.setBounds(470, 15, 142, 30);
        bankCardLabel.setFont(new Font("Helvetica", Font.BOLD, 25));
        bankCardLabel.setForeground(Color.WHITE);
        panel.add(bankCardLabel);

        JLabel wBankCardJLabel = new JLabel("Thank You For Choosing Us");
        wBankCardJLabel.setBounds(385, 55, 411, 30);
        wBankCardJLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
        wBankCardJLabel.setForeground(Color.WHITE);
        panel.add(wBankCardJLabel);


        ///////////////////--------------------------------------Debit Card Details------------------------------------------------///////////////
        debitCardDetailsLabel = new JLabel("DebitCard Details");
        debitCardDetailsLabel.setBounds(135, 110, 250, 30);
        debitCardDetailsLabel.setForeground(new Color(1, 141, 177));
        debitCardDetailsLabel.setFont(new Font("Helvetica", Font.BOLD, 25));
        frame.add(debitCardDetailsLabel);


        //////////////////Debit card details labels//////////////////
        balanceAmountLabel = new JLabel("Balance Amount: ");
```

```
balanceAmountLabel.setBounds(85, 162, 132, 24);
frame.add(balanceAmountLabel);

cardIdLabel = new JLabel("Card ID: ");
cardIdLabel.setBounds(85, 195, 132, 24);
frame.add(cardIdLabel);

bankAccountLabel = new JLabel("Bank Account: ");
bankAccountLabel.setBounds(85, 228, 132, 24);
frame.add(bankAccountLabel);

issuerBankLabel = new JLabel("Issuer Bank: ");
issuerBankLabel.setBounds(85, 261, 132, 24);
frame.add(issuerBankLabel);

clientNameLabel = new JLabel("Client Name: ");
clientNameLabel.setBounds(85, 295 ,132, 24);
frame.add(clientNameLabel);

pinNumberLabel = new JLabel("PIN Number: ");
pinNumberLabel.setBounds(85, 327 ,132, 24);
frame.add(pinNumberLabel);



////////////////////////debit card details text field//////////////////
balanceAmountTextField = new JTextField();
balanceAmountTextField.setBounds(214, 162, 234, 25);
frame.add(balanceAmountTextField);
```

```
cardIdTextField = new JTextField();
cardIdTextField.setBounds(214, 195, 234, 25);
frame.add(cardIdTextField);

bankAccountTextField = new JTextField();
bankAccountTextField.setBounds(214, 228, 234, 25);
frame.add(bankAccountTextField);

issuerBankTextField = new JTextField();
issuerBankTextField.setBounds(214, 261, 234, 25);
frame.add(issuerBankTextField);

clientNameTextField = new JTextField();
clientNameTextField.setBounds(214, 295, 234, 25);
frame.add(clientNameTextField);

pinNumberTextField = new JTextField();
pinNumberTextField.setBounds(214, 327, 234, 25);
frame.add(pinNumberTextField);



////////////////add debitcard button to add the details of debitcard////////////////
addDebitCardButton = new JButton("Add DebitCard");
addDebitCardButton.setBounds(100, 353, 145, 30);
addDebitCardButton.setFont(new Font("Arial", Font.PLAIN, 16));
addDebitCardButton.setBackground(new Color(1, 141, 177));
addDebitCardButton.setForeground(Color.WHITE);
```

```
        addDebitCardButton.setFocusPainted(false);
        addDebitCardButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
        addDebitCardButton.setPreferredSize(new Dimension(120, 40));
        frame.add(addDebitCardButton);

        ////////////////display button to display the details of debitcard////////////////
        displayDebitCardButton = new JButton("Display Debit");
        displayDebitCardButton.setBounds(275, 353, 145, 30);
        displayDebitCardButton.setFont(new Font("Arial", Font.PLAIN, 16));
        displayDebitCardButton.setBackground(new Color(1, 141, 177));
        displayDebitCardButton.setForeground(Color.WHITE);
        displayDebitCardButton.setFocusPainted(false);
        displayDebitCardButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10,
20));
        displayDebitCardButton.setPreferredSize(new Dimension(120, 40));
        frame.add(displayDebitCardButton);



        ///////////////////witdraw from debitcard/////////////////
        debitCardLabel = new JLabel("Withdraw from DebitCard");
        debitCardLabel.setBounds(120, 510, 316, 30);
        debitCardLabel.setForeground(new Color(1, 141, 177));
        debitCardLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
        frame.add(debitCardLabel);

        /////////////////withdraw from debit card label///////////////////
        cardId3Label = new JLabel("Card ID: ");
        cardId3Label.setBounds(85, 555, 132, 24);
```

```
frame.add(cardId3Label);

withdrawalAmountLabel = new JLabel("Withdrawal Amount: ");
withdrawalAmountLabel.setBounds(85, 588, 132, 24);
frame.add(withdrawalAmountLabel);

pinNumber1Label = new JLabel("PIN Number: ");
pinNumber1Label.setBounds(85, 621, 132, 24);
frame.add(pinNumber1Label);

dateOfWithdrawalLabel = new JLabel("Date of Withdrawal: ");
dateOfWithdrawalLabel.setBounds(85, 654, 132, 24);
frame.add(dateOfWithdrawalLabel);



/////////////////withdraw from debit card textfield///////////////////
cardId3TextField = new JTextField();
cardId3TextField.setBounds(214, 555, 234, 25);
frame.add(cardId3TextField);

withdrawalAmountTextField = new JTextField();
withdrawalAmountTextField.setBounds(214, 588, 234, 25);
frame.add(withdrawalAmountTextField);

pinNumber1TextField = new JTextField();
pinNumber1TextField.setBounds(214, 621, 234, 25);
frame.add(pinNumber1TextField);
```

```
    /////////////////////date of withdrawal of debit card///////////////////

    Integer[] day = new Integer[31];
    for(int i=0; i<31; i++){
       day[i] = i + 1;
    }
    debitDayComboBox = new JComboBox(day);
    debitDayComboBox.setBounds(214, 654, 60, 30);
    frame.add(debitDayComboBox);

    String
month[]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
    debitMonthComboBox = new JComboBox(month);
    debitMonthComboBox.setBounds(279, 654, 60, 30);
    frame.add(debitMonthComboBox);

    String year[]={"2023","2024","2025","2026"};
    debitYearComboBox = new JComboBox(year);
    debitYearComboBox.setBounds(344, 654, 60, 30);
    frame.add(debitYearComboBox);



    ////////////////withdraw Button for debitcard////////////////
    withdrawButton = new JButton("Withdraw");
    withdrawButton.setBounds(214, 687, 234, 30);
    withdrawButton.setFont(new Font("Arial", Font.PLAIN, 16));
    withdrawButton.setBackground(new Color(1, 141, 177));
```

```
        withdrawButton.setForeground(Color.WHITE);
        withdrawButton.setFocusPainted(false);
        withdrawButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
        withdrawButton.setPreferredSize(new Dimension(120, 40));
        frame.add(withdrawButton);



        //////////////////-----------------------------------Credit Card Details-----------------------------
////////////////
        creditCardDetailsLabel = new JLabel("CreditCard Details");
        creditCardDetailsLabel.setBounds(685, 110, 250, 30);
        creditCardDetailsLabel.setForeground(new Color(1, 141, 177));
        creditCardDetailsLabel.setFont(new Font("Helvetica", Font.BOLD, 25));
        frame.add(creditCardDetailsLabel);



        ////////////////////////////////////////credit card details labebls//////////////////////////
        balanceAmount1Label = new JLabel("Balance Amount: ");
        balanceAmount1Label.setBounds(620, 162, 132, 24);
        frame.add(balanceAmount1Label);

        cardId1Label = new JLabel("Card ID: ");
        cardId1Label.setBounds(620, 195, 132, 24);
        frame.add(cardId1Label);

        bankAccount1Label = new JLabel("Bank Account: ");
        bankAccount1Label.setBounds(620, 228, 132, 24);
        frame.add(bankAccount1Label);
```

```
issuerBank1Label = new JLabel("Issuer Bank: ");
issuerBank1Label.setBounds(620, 261, 132, 24);
frame.add(issuerBank1Label);

clientName1Label = new JLabel("Client Name: ");
clientName1Label.setBounds(620, 295 ,132, 24);
frame.add(clientName1Label);

cvcNumberLabel = new JLabel("CVC Number: ");
cvcNumberLabel.setBounds(620, 327, 132, 24);
frame.add(cvcNumberLabel);

interestRateLabel = new JLabel("Interest Rate: ");
interestRateLabel.setBounds(620, 360, 132, 24);
frame.add(interestRateLabel);

expirationDateLabel = new JLabel("Expiration Date: ");
expirationDateLabel.setBounds(620, 393, 132, 24);
frame.add(expirationDateLabel);



//////////////////credit card details text field/////////////////////////
balanceAmount1TextField = new JTextField();
balanceAmount1TextField.setBounds(755, 162, 234, 25);
frame.add(balanceAmount1TextField);

cardId1TextField = new JTextField();
```

```
cardId1TextField.setBounds(755, 195, 234, 25);
frame.add(cardId1TextField);

bankAccount1TextField = new JTextField();
bankAccount1TextField.setBounds(755, 228, 234, 25);
frame.add(bankAccount1TextField);

issuerBank1TextField = new JTextField();
issuerBank1TextField.setBounds(755, 261, 234, 25);
frame.add(issuerBank1TextField);

clientName1TextField = new JTextField();
clientName1TextField.setBounds(755, 295, 234, 25);
frame.add(clientName1TextField);

cvcNumberTextField = new JTextField();
cvcNumberTextField.setBounds(755, 327, 234, 25);
frame.add(cvcNumberTextField);

interestRateTextField = new JTextField();
interestRateTextField.setBounds(755, 360, 234, 25);
frame.add(interestRateTextField);



///////////////////expiration date of credit card///////////////////////
Integer[] day1 = new Integer[31];
for(int i=0; i<31; i++){
   day1[i] = i + 1;
```

```
        }
        dayComboBox = new JComboBox(day1);
        dayComboBox.setBounds(755, 393, 60, 30);
        frame.add(dayComboBox);



        String
month1[]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
        monthComboBox = new JComboBox(month1);
        monthComboBox.setBounds(820, 393, 60, 30);
        frame.add(monthComboBox);

        String year1[]={"2023","2024","2025","2026"};
        yearComboBox = new JComboBox(year);
        yearComboBox.setBounds(885, 393, 60, 30);
        frame.add(yearComboBox);



        ///////////////////////////////button to add credit card details//////////////////
        addCreditCardButton = new JButton("Add CreditCard");
        addCreditCardButton.setBounds(635, 426, 155, 30);
        addCreditCardButton.setFont(new Font("Arial", Font.PLAIN, 16));
        addCreditCardButton.setBackground(new Color(1, 141, 177));
        addCreditCardButton.setForeground(Color.WHITE);
        addCreditCardButton.setFocusPainted(false);
        addCreditCardButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10,
20));
        addCreditCardButton.setPreferredSize(new Dimension(120, 40));
        frame.add(addCreditCardButton);
```

```
    /////////////////button to display credit card details//////////////////////////////
    displayCreditCardButton = new JButton("Display Credit");
    displayCreditCardButton.setBounds(820, 426, 145, 30);
    displayCreditCardButton.setFont(new Font("Arial", Font.PLAIN, 16));
    displayCreditCardButton.setBackground(new Color(1, 141, 177));
    displayCreditCardButton.setForeground(Color.WHITE);
    displayCreditCardButton.setFocusPainted(false);
    displayCreditCardButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10,
20));
    displayCreditCardButton.setPreferredSize(new Dimension(120, 40));
    frame.add(displayCreditCardButton);



    ////////////////////setcredit limit of credit card label////////////////////////


    creditCardLabel = new JLabel("Set Credit Limit");
    creditCardLabel.setBounds(720, 510, 250, 30);
    creditCardLabel.setForeground(new Color(1, 141, 177));
    creditCardLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
    frame.add(creditCardLabel);

    //-------------setcredit limit----------label-----////
    cardId4Label = new JLabel("Card ID: ");
    cardId4Label.setBounds(620, 555, 132, 24);
    frame.add(cardId4Label);

    setCreditLimitLabel = new JLabel("Set Credit Limit: ");
```

```
setCreditLimitLabel.setBounds(620, 588, 132, 24);
frame.add(setCreditLimitLabel);

gracePeriodLabel = new JLabel("Grace Period: ");
gracePeriodLabel.setBounds(620, 621, 132, 24);
frame.add(gracePeriodLabel);

//--------------set credit limit--------textfield//
cardId4TextField = new JTextField();
cardId4TextField.setBounds(755, 555, 234, 25);
frame.add(cardId4TextField);

setCreditLimitTextField = new JTextField();
setCreditLimitTextField.setBounds(755, 588, 234, 25);
frame.add(setCreditLimitTextField);

gracePeriodTextField = new JTextField();
gracePeriodTextField.setBounds(755, 621, 234, 25);
frame.add(gracePeriodTextField);


//////////////button to set credit limit////////////////////
setCreditLimitButton = new JButton("Set Credit Limit");
setCreditLimitButton.setBounds(755, 654, 234, 30);
setCreditLimitButton.setFont(new Font("Arial", Font.PLAIN, 16));
setCreditLimitButton.setBackground(new Color(1, 141, 177));
setCreditLimitButton.setForeground(Color.WHITE);
setCreditLimitButton.setFocusPainted(false);
setCreditLimitButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
setCreditLimitButton.setPreferredSize(new Dimension(120, 40));
frame.add(setCreditLimitButton);

///////////////button to cancel credit///////////////////////////////
cancelCreditButton = new JButton("Cancel Credit");
cancelCreditButton.setBounds(755, 687, 234, 30);
cancelCreditButton.setFont(new Font("Arial", Font.PLAIN, 16));
cancelCreditButton.setBackground(new Color(1, 141, 177));
cancelCreditButton.setForeground(Color.WHITE);
cancelCreditButton.setFocusPainted(false);
cancelCreditButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
cancelCreditButton.setPreferredSize(new Dimension(120, 40));
frame.add(cancelCreditButton);


///////////////////////////////clear button///////////////////////////////
clearButton = new JButton("Clear");
clearButton.setBounds(470, 720, 180, 30);
clearButton.setFont(new Font("Arial", Font.PLAIN, 16));
clearButton.setBackground(new Color(1, 141, 177));
clearButton.setForeground(Color.WHITE);
clearButton.setFocusPainted(false);
clearButton.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
clearButton.setPreferredSize(new Dimension(120, 40));
frame.add(clearButton);
```

```
    ///////////////action listeners//////////////////////////////
    addDebitCardButton.addActionListener(this);
    addCreditCardButton.addActionListener(this);
    displayDebitCardButton.addActionListener(this);
    displayCreditCardButton.addActionListener(this);
    withdrawButton.addActionListener(this);
    setCreditLimitButton.addActionListener(this);
    cancelCreditButton.addActionListener(this);
    clearButton.addActionListener(this);



    frame.setTitle("Bank Card");

    frame.setLayout(null);

    frame.setSize(1100, 800);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.setResizable(false);

    frame.setVisible(true);
  }
  public static void main(String[] args){
    BankCardGUI GUI = new  BankCardGUI();
  }

  public void actionPerformed(ActionEvent e){
    //write the logic of the button functionality here
    if(e.getSource() == addDebitCardButton){
        try{
            // Get balance amount from text field
            String balanceAmount = balanceAmountTextField.getText();
            int balanceAmountValue = Integer.parseInt(balanceAmount);

            // Get card ID from text field
            String cardId = cardIdTextField.getText();
            int cardIdValue = Integer.parseInt(cardId);

            // Get bank account from text field
            String bankAccount = bankAccountTextField.getText();
            // Get issuer bank from text field
            String issuerBank = issuerBankTextField.getText();
            // Get issuer bank from text field
            String clientName = clientNameTextField.getText();
            // Get PIN number from text field

            String pinNumber = pinNumberTextField.getText();
            int pinNumberValue = Integer.parseInt(pinNumber);

            // Create a new DebitCard object with the extracted values
            DebitCard debitCardDetails = new DebitCard (balanceAmountValue,
cardIdValue, bankAccount, issuerBank, clientName, pinNumberValue);

            if(cards.isEmpty()){
                // If the cards list is empty, add the new debit card directly
                cards.add(debitCardDetails);
                JOptionPane.showMessageDialog(frame, "DebitCard has been added",
"Success", JOptionPane.INFORMATION_MESSAGE);


            }
            else{
                boolean isAdded = true;
                for(BankCard storeDebitCards: cards){
                    if(cardIdValue == storeDebitCards.getCardId()){
                        // If a card with the same card ID exists, show an error message
                        JOptionPane.showMessageDialog(frame, "This DebitCard has already
been added", "Alert", JOptionPane.ERROR_MESSAGE);
                        isAdded = false;
                        break; //Exit the loop once a matching card id is found
                    }
                    else{
                        isAdded = true;
                    }
                }


                if(isAdded == true){
                    cards.add(debitCardDetails);
                    JOptionPane.showMessageDialog(frame, "New DebitCard has been
added", "Success", JOptionPane.INFORMATION_MESSAGE);
                }
            }
        }
        catch(NumberFormatException nfe){
             // Show a warning message if input information is invalid
```

```
                JOptionPane.showMessageDialog(frame, "The information you have input
cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);
            }
        }



        if(e.getSource() == addCreditCardButton){
            try{
                // Get balance amount from text field
                String balanceAmount = balanceAmount1TextField.getText();
                double balanceAmount1Value = Double.parseDouble(balanceAmount);

                // Get card ID from text field
                String cardId1 = cardId1TextField.getText();
                int cardId1Value = Integer.parseInt(cardId1);

                // Get bank account from text field
                String bankAccount = bankAccount1TextField.getText();
                // Get issuer bank from text field
                String issuerBank = issuerBank1TextField.getText();
                // Get client name from text field
                String clientName = clientName1TextField.getText();

                // Get CVC number from text field
                String cvcNumber = cvcNumberTextField.getText();
                int cvcNumberValue = Integer.parseInt(cvcNumber);

                // Get interest rate from text field
                String interestRate = interestRateTextField.getText();
                double interestRateValue = Double.parseDouble(interestRate);



                // Get interest rate from text field
                String year  = (String)yearComboBox.getSelectedItem();
                String month = (String)monthComboBox.getSelectedItem();
                int day = (int)dayComboBox.getSelectedItem();
                String dayString = String.valueOf(day);

                String expirationDate = (year +","+ month +" "+ dayString);



                // Create a new CreditCard object with the extracted values
                CreditCard creditCardDetails = new CreditCard (balanceAmount1Value,
cardId1Value, bankAccount, issuerBank, clientName, cvcNumberValue,
interestRateValue, expirationDate);


                if(cards.isEmpty()){
                  // If the cards list is empty, add the new credit card directly
                  cards.add(creditCardDetails);
                  JOptionPane.showMessageDialog(frame, "CreditCard has been added",
"Success", JOptionPane.INFORMATION_MESSAGE);
                }
                else{
                  boolean isAdded = false;
                  for(BankCard storeCreditCards: cards){
                    if(cardId1Value == storeCreditCards.getCardId()){
                      // If a card with the same card ID exists, show an error message
```

```
                    JOptionPane.showMessageDialog(frame, "CreditCard has already
been added", "Alert", JOptionPane.ERROR_MESSAGE);
                    isAdded = false;
                    break; //Exit the loop once a matching card id is found
                }
                else{
                     // If no card with the same card ID exists, set isAdded to true
                    isAdded = true;
                }
            }


            if(isAdded == true){
                // If isAdded is true, add the new credit card
                cards.add(creditCardDetails);
                JOptionPane.showMessageDialog(frame, "New CreditCard has been
added", "Success", JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
    catch(NumberFormatException nfe){
        // Show a warning message if input information is invalid
        JOptionPane.showMessageDialog(frame, "The information you have input
cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);


    }
}



if (e.getSource() == displayDebitCardButton) {
```

```
            // Variable to track if a debit card is found
            boolean foundDebitCard = false;

            // Iterate through the cards list to find debit cards
            for (BankCard storeDebitCards : cards) {
                if (storeDebitCards instanceof DebitCard) {
                    System.out.println("************************************************\n");
                    ((DebitCard) storeDebitCards).display();
                    System.out.println("\n");
                    foundDebitCard = true;
                }
            }
            if (foundDebitCard) {
                JOptionPane.showMessageDialog(frame, "The details have been displayed",
"Information", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(frame, "No DebitCard found",
"Information", JOptionPane.INFORMATION_MESSAGE);
            }
        }

        if (e.getSource() == displayCreditCardButton) {
            // Variable to track if a credit card is found
            boolean foundCreditCard = false;

            // Iterate through the cards list to find credit cards
            for (BankCard storeCreditCards : cards) {
                if (storeCreditCards instanceof CreditCard) {
                    System.out.println("************************************************\n");
                    ((CreditCard) storeCreditCards).display();
                    System.out.println("\n");
                    foundCreditCard = true;
                }
            }
            if (foundCreditCard) {
                JOptionPane.showMessageDialog(frame, "The details have been displayed",
"Information", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(frame, "No CreditCard found",
"Information", JOptionPane.INFORMATION_MESSAGE);
            }
        }



        if (e.getSource() == withdrawButton) {
            try {
                // Get card ID from text field
                String cardId3 = cardId3TextField.getText();
                int cardId3Value = Integer.parseInt(cardId3);

                // Get PIN number from text field
                String pinNumberWithdraw = pinNumber1TextField.getText();
                int pinNumberWithdrawValue = Integer.parseInt(pinNumberWithdraw);

                // Get withdrawal amount from text field
                String withdrawalAmount = withdrawalAmountTextField.getText();
                int withdrawalValue = Integer.parseInt(withdrawalAmount);
                // Get selected values from combo boxes for the withdrawal date
                String year = (String) debitYearComboBox.getSelectedItem();
                String month = (String) debitMonthComboBox.getSelectedItem();
                int day1 = (int) dayComboBox.getSelectedItem();
                String day1String = String.valueOf(day1);

                // Join the selected values to create the withdrawal date string
                String dateOfWithdrawal = year +","+ month +" "+ day1String;

                // Variable to track if a debit card is found
                boolean debitcard = false;

                if (cards.isEmpty()) {
                    // If the cards list is empty, show an error message
                    JOptionPane.showMessageDialog(frame, "Cannot Withdraw, DebitCard
has not been added yet", "Alert", JOptionPane.ERROR_MESSAGE);
                } else {
                    for (BankCard withdrawCards : cards) {
                        if (withdrawCards instanceof DebitCard) {
                            if (cardId3Value == withdrawCards.getCardId()) {
                                debitcard = true;
                                if (pinNumberWithdrawValue == ((DebitCard)
withdrawCards).getPinNumber()) {
                            if (((DebitCard) withdrawCards).checkBalance(withdrawalValue))
{
                                ((DebitCard)
withdrawCards).withdraw(pinNumberWithdrawValue, withdrawalValue,
dateOfWithdrawal);
                                JOptionPane.showMessageDialog(frame, "The amount has
been withdrawn successfully", "Success", JOptionPane.INFORMATION_MESSAGE);
```

```
                                    break;
                                } else {
                                    JOptionPane.showMessageDialog(frame, "Insufficient
balance", "Success", JOptionPane.INFORMATION_MESSAGE);
                                }
                            } else {
                                JOptionPane.showMessageDialog(frame, "Incorrect PIN
number", "Alert", JOptionPane.ERROR_MESSAGE);
                            }
                        } else {
                            debitcard = false;
                        }
                    }
                }
            }

            if (debitcard == false) {
                JOptionPane.showMessageDialog(frame, "The DebitCard with the provided
ID has not been found", "Alert", JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException nfex) {
            JOptionPane.showMessageDialog(frame, "The information you provided
cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);
        }
    }


    if (e.getSource() == setCreditLimitButton) {
        try {
```

```
        // Get card ID from text field
        String cardId4 = cardId4TextField.getText();
        int cardIdCreditLimit = Integer.parseInt(cardId4);

        // Get credit limit value from text field
        String setCreditLimit = setCreditLimitTextField.getText();
        int setCreditLimitValue = Integer.parseInt(setCreditLimit);

        // Get grace period value from text field
        String gracePeriod = gracePeriodTextField.getText();
        int gracePeriodValue = Integer.parseInt(gracePeriod);

        // Variable to track if a credit card is found
        boolean creditcard = false;

        if (cards.isEmpty()) {
            // If the cards list is empty, show an error message
            JOptionPane.showMessageDialog(frame, "Cannot set credit limit.
CreditCard has not been added yet", "Alert", JOptionPane.ERROR_MESSAGE);
        } else {
            for (BankCard creditCards : cards) {
                if (creditCards instanceof CreditCard) {
                    if (cardIdCreditLimit == creditCards.getCardId()) {
                        creditcard = true;
                        if (setCreditLimitValue <= 2.5 * creditCards.getBalanceAmount()) {
                            ((CreditCard) creditCards).setCreditLimit(setCreditLimitValue,
gracePeriodValue);
                            JOptionPane.showMessageDialog(frame, "The credit limit has
been set successfully", "Success", JOptionPane.INFORMATION_MESSAGE);
```

```
                            break;
                        } else {
                            JOptionPane.showMessageDialog(frame, "The amount exceeds the credit limit", "Alert", JOptionPane.ERROR_MESSAGE);
                        }
                    } else {
                        creditcard = false;
                    }
                }
            }
        }

        if (creditcard == false) {
            JOptionPane.showMessageDialog(frame, "The Creditcard with the provided ID has not been found", "Alert", JOptionPane.ERROR_MESSAGE);
        }
    } catch (NumberFormatException nfex) {
        JOptionPane.showMessageDialog(frame, "The information you provided cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);
    }
}



if (e.getSource() == cancelCreditButton) {
    try {
        // Get card ID from text field
        String cardId4 = cardId4TextField.getText();
        int cardIdCreditLimit = Integer.parseInt(cardId4);
```

```
            // Variable to track if the credit card is found
            boolean creditlimit = false;


            if (cards.isEmpty()) {
                // If the cards list is empty, show an error message
                JOptionPane.showMessageDialog(frame, "Cannot cancel credit card.
CreditCard has not been added yet", "Alert", JOptionPane.ERROR_MESSAGE);
            } else {
                for (BankCard creditCards : cards) {
                    if (creditCards instanceof CreditCard) {
                        if (cardIdCreditLimit == creditCards.getCardId()) {
                            // Cancel the credit card
                            ((CreditCard) creditCards).cancelCreditCard();
                            JOptionPane.showMessageDialog(frame, "The CreditCard has
been canceled", "Success", JOptionPane.INFORMATION_MESSAGE);
                            creditlimit = true;
                            break;
                        }
                    }
                }
            }
            if(creditlimit == false){
                JOptionPane.showMessageDialog(frame, "The card ID provided does not
exist", "Alert", JOptionPane.ERROR_MESSAGE);


            }
        }
        catch (NumberFormatException nfex) {
            JOptionPane.showMessageDialog(frame, "The information you provided
cannot be accepted", "Alert", JOptionPane.WARNING_MESSAGE);
        }
    }



    if(e.getSource() == clearButton){
        // Clear the text fields by setting their text to empty strings
        balanceAmountTextField.setText("");
        cardIdTextField.setText("");
        bankAccountTextField.setText("");
        issuerBankTextField.setText("");
        clientNameTextField.setText("");
        pinNumberTextField.setText("");
        balanceAmount1TextField.setText("");
        cardId1TextField.setText("");
        bankAccount1TextField.setText("");
        issuerBank1TextField.setText("");
        clientName1TextField.setText("");
        cvcNumberTextField.setText("");
        interestRateTextField.setText("");
        pinNumber1TextField.setText("");
        cardId3TextField.setText("");
        withdrawalAmountTextField.setText("");
        pinNumberTextField.setText("");
        cardId4TextField.setText("");
        setCreditLimitTextField.setText("");
        gracePeriodTextField.setText("");
    }
}

}
```

Code of the BankCard class

```
public class BankCard

{

    //attributes //initializing the variables

    private int cardId;

    private String clientName;

    private String issuerBank;

    private String bankAccount;

    private double balanceAmount;




    /**constructor for BankCard super class*/

    public BankCard(double balanceAmount, int cardId, String bankAccount, String
issuerBank)

    {

        this.cardId = cardId;

        this.issuerBank = issuerBank;

        this.bankAccount = bankAccount;

        this.balanceAmount = balanceAmount;

        this.clientName = "";

    }
```

```
//getter methods for all the variables //accessor
 public int getCardId()
{
   return this.cardId;
}

public String getClientName()
{
   return this.clientName;
}

public String getIssuerBank()
{
   return this.issuerBank;
}

public String getBankAccount()
{
   return this.bankAccount;
}
```

```
public double getBalanceAmount()

{

    return this.balanceAmount;

}





//setter method for the variables //mutator

public void setClientName(String clientName)

{

    //attribute = parameter

    this.clientName = clientName;

}


public void setBalanceAmount(double balanceAmount)

{

    this.balanceAmount = balanceAmount;

}


//checking whether balance is sufficient for withdrawal

//checkbalance method

public boolean checkBalance(int withdrawalAmount)
```

```
{
    return this.balanceAmount >= withdrawalAmount;
}




/**method to display the details*/
 /*
  This is the display method of the super class.
 */
public void display() //Display method
{
System.out.println("Card ID: " + cardId);
if(clientName.equals("")) {
    System.out.println("Client name: not assigned");
}
else {
    System.out.println("Client name: " + clientName);
}
System.out.println("Issuer bank: " + issuerBank);
System.out.println("Bank account: " + bankAccount);
```

```
    System.out.println("Balance amount: " + balanceAmount);

    }

}
```

Code for DebitCard Class

```
public class DebitCard extends BankCard

{

   //attributes //initializing the variables

   private int pinNumber;

   private int withdrawalAmount;

   private String dateOfWithdrawal;

   private boolean hasWithdrawn;


    /**constructor for DebitCard subclass*/

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

   {

      super(balanceAmount, cardId, bankAccount, issuerBank);

      super.setClientName(clientName);

      this.pinNumber = pinNumber;

      this.withdrawalAmount = 0;

      this.dateOfWithdrawal = "";

      this.hasWithdrawn = false;

   }



   //getter methods for all the variables //accessor
```

```
public int getPinNumber()

{

   return this.pinNumber;

}


public int getWithdrawalAmount()

{

   return this.withdrawalAmount;

}


public String getDateOfWithdrawal()

{

   return this.dateOfWithdrawal;

}


public boolean getHasWithdrawn()

{

   return this.hasWithdrawn;

}



//setter method for the variables //mutator
```

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




/**withdraw method*/
/*
 This is a method to check the withdrawal which accepts three parameters.
*/
public void withdraw(int pinNumber, int withdrawalAmount, String dateOfWithdrawal)
{
    if(this.pinNumber == pinNumber)
    {
    System.out.println("Pin is valid");
        if(super.checkBalance(withdrawalAmount))
        {
        this.withdrawalAmount = withdrawalAmount;
        this.dateOfWithdrawal = dateOfWithdrawal;
        this.hasWithdrawn = true;
        super.setBalanceAmount  (super.getBalanceAmount()  -  withdrawalAmount);//  a
new value is assign to the balanceAmount with setter method of the super class after the
withdrawal//

        System.out.println("Withdrawal has been made");

        }

        else

        {

        System.out.println("Insufficient Balance amount");

      }

  }

  else

  {


        System.out.println("Invalid Pin number");

  }

  }





  /**method to display the results*/

  /*

   This is the display method of the DebitCard subclass.

   It also calls the display method of the super class.
```

```
    */
    public void display() //Display method
    {
        super.display();
        System.out.println("PIN number: " + pinNumber);
        if(hasWithdrawn)
        {
            System.out.println("Withdrawal amount: " + withdrawalAmount);
            System.out.println("Date of Withdrawal: " + dateOfWithdrawal);
        }
        else
        {
            System.out.println("No withdrawal has been made.");
        }
    }
}
```

Code for CreditCard Class

```
public class CreditCard extends BankCard

{

    //attributes //initializing the variables

    private int cvcNumber;

    private double creditLimit;

    private double interestRate;

    private String expirationDate;

    private int gracePeriod;

    private boolean isGranted;



    /**constructor for CreditCard subclass*/

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

    {

        super(balanceAmount, cardId, bankAccount, issuerBank);

        super.setClientName(clientName);

        this.cvcNumber = cvcNumber;

        this.interestRate = interestRate;

        this.expirationDate = expirationDate;

        this.isGranted = false;
```

```
}



//getter methods for all the variables //accessor
public int getCvcNumber()
{
    return this.cvcNumber;
}

public double getCreditLimit()
{
    return this.creditLimit;
}

public double getInterestRate()
{
    return this.interestRate;
}

public String getExpirarionDate()
{
    return this.expirationDate;
```

```
}


public int getGracePeriod()
{
    return this.gracePeriod;
}


public boolean getIsGranted()
{
    return this.isGranted;
}



//setter method for the variables //mutator
public void setCreditLimit(int creditLimit, int gracePeriod)
{
    if(creditLimit <= 2.5 * super.getBalanceAmount())
    {
        this.creditLimit = creditLimit;
        this.gracePeriod = gracePeriod;
        this.isGranted = true;
        System.out.println("Credit is granted.");
    }
    else
    {
        System.out.println("Credit cannot be issued.");
    }
}

/**cancle credit card method*/
/*
This method cancels the credit card and reset all the value of the variables to their default values
and displays a suitable message
*/
public void cancelCreditCard()
{
    this.cvcNumber = 0;
    this.creditLimit = 0;
    this.interestRate = 0;
    this.expirationDate = "";
    this.gracePeriod = 0;
    this.isGranted = false;
    System.out.println("Credit card has been canceled.");
}




/**method to display the results*/

/*

This is the display method of the CreditCard subclass.

*/

public void display(){

    super.display();

    if(isGranted){

        System.out.println("CVC number: " + cvcNumber);

        System.out.println("Credit Limit: " + creditLimit);

        System.out.println("Interest rate: " + interestRate);

        System.out.println("Expiration date: " + expirationDate);

        System.out.println("Grace period: " + gracePeriod);

    }

    else{

        System.out.println("Credit card has not been granted yet.");

    }

}

}
```