CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Samir Gurung

London Met ID: 22015816

College ID: NP01CP4A220131

Assignment Due Date: Wednesday, May 10, 2023

Assignment Submission Date: Wednesday, May 10, 2023

Word Count: 6423

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am tfully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Contents

Table of Figures

Table of Tables

1. Introduction

This was the second course work that we got, to understand the concept of java more precisely by adding new class BankGUI. The previous assignment was about knowing the real world problem scenario using the object-oriented concept of java on Banking sector. We created different classes to represent a Bank card, with two subclasses Debit card and Credit card on the previous course work. On this assignment, we had to make a graphical user interface (GUI) for a system that stores details of Bank card in an arraylist. All together we have four classes- Bank card, Debit card, Credit card and BankGUI. By using all these classes we should be able to make a graphical user interface (GUI) for a system that stores all the details of Bank Card.

On our new class Bank GUI different new components have been added through bluej. The class contains various java components to construct it and to make it look good. The code has functions for adding different classes to an array list, performing different actions on the cards, and updating the cards' data as required. It also offers guidelines for handling errors with try catch method, managing events with interfaces and frameworks, and more. Components like JLabel, JTextfields, JButtons etc are frequently used while constructing the GUI. I also used Software like moqups to make wireframes just to make it easy and for better understanding while working on GUI through coding. The course work also includes concepts on event management, exception handling, object oriented programming inheritance and GUI based programs . Hence, the report has been made thoroughly by learning many programming process and components for making a BankGUI.

1.2. Tools Used

1.2.1. BlueJ

The main purpose of BlueJ's creation was to aid in user education on object-oriented programming. It is a windows-based platform for Java Developing Kit. It was created to assist in the teaching and learning of OOPs (object-oriented programming). The interface allows class and coded object visual views. Compared to the interface of the bulk of popular IDEs, the BlueJ user interface is simpler. There are numerous technologies available that are tailored to its learning objectives. In addition, tools for industry development such an editor, compiler, and runtime environment are accessible.

1.2.2. Ms Word

MS Word is a word processing program that was first developed by Microsoft in 1983. A variety of pre-set styles and designs are offered by Microsoft Word, making it simple to format large documents with a single click. Moreover, you may draw shapes, create and add a range of charts, and insert images and videos from your computer and the internet. There are features that enable you to quickly create a table of contents. Together with headers and footers, footnotes can be inserted. There are options for creating a table of figures, a bibliography, and even cross references.

2. Class Diagram

A class diagram is a software used in scheming and modelling a diagram to explain class and their relations. It is a blueprint of system or subsystem. The class diagram is one of the UML diagram types that is used to depict static diagrams. It uses classes, attributes, relations, and operations amongst various objects to represent system structure. By using class diagrams, we can create software at a high level without having to look at the source code. The characteristics and functions of a class are described in a class diagram, along with the restrictions placed on the system.

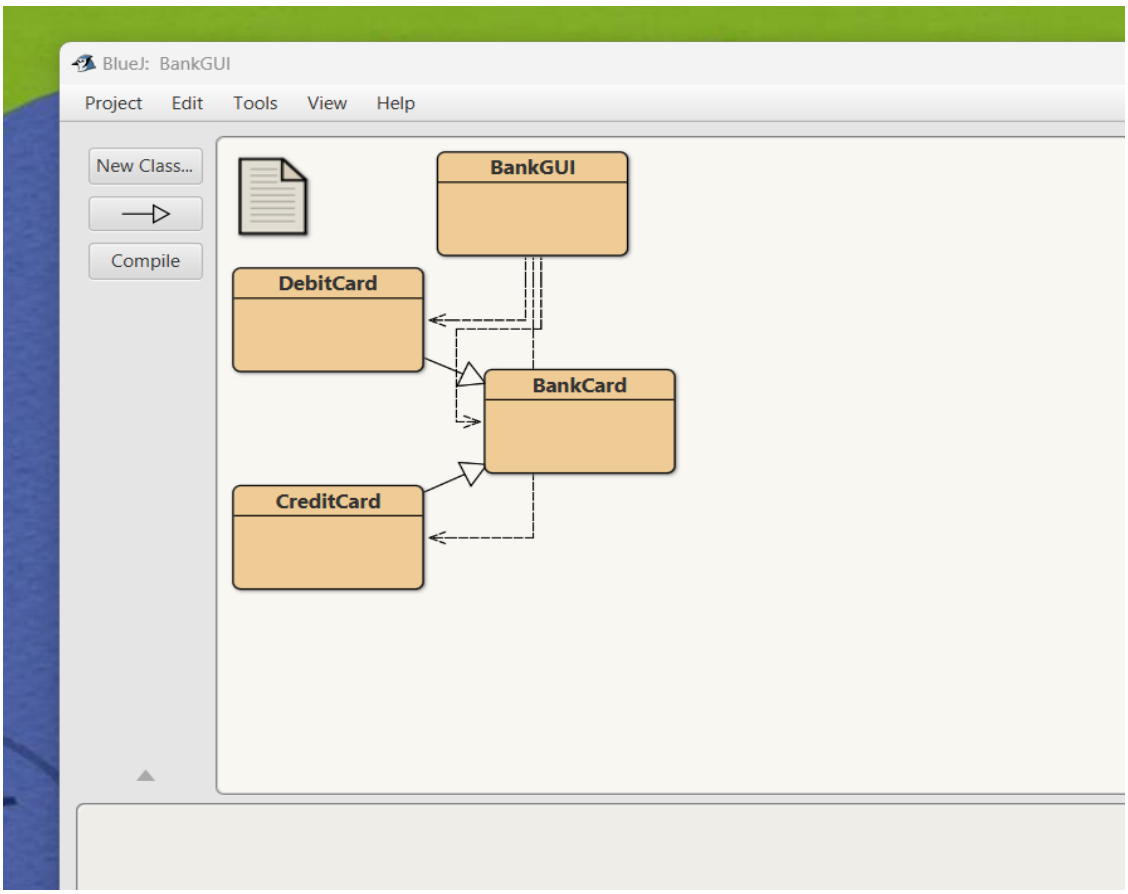

Figure 1 : Class in BlueJ

2.1. BankCard

Bank Card

- cardID: int
- clientName: String
- issuerBank: String
- bankAccount: String
- balanceAmount: double

+ BankCard(balanceAmount: double, cardID: int, bankAccount: String, issuerBank: String)
+ getClientName(): String
+ getBalanceAmount(): double
+ getCardID(): int
+ getBankAccount(): String
+ getIssuerBank(): String
+ setClientName(clientName: String): void
+ setBalanceAmount(balanceAmount: double): void
+ display(): void

Figure 2 ; Bank Card Class

2.2. DebitCard

Debit Card
- pinNumber: int - withdrawalAmount: int - dateOfWithdrawal: String - hasWithdrawn: boolean
+ DebitCard(balanceAmount: double, cardID: int, bankAccount: String, clientName: String, pinNumber: int) + getPinNumber(): int + getWithdrawalAmount(): int + getDateOfWithdrawal(): String + getHasWithdrawn(): boolean + setWithdrawalAmount(withdrawalAmount: int): void + withdraw(withdrawalAmount: int, dateOfWithdrawal: String, pinNumber: int): void + display(): void

Figure 3 ; Debit Card Class

2.3. CreditCard

Credit Card
- cvcNumber: int - creditLimit: double - interestRate: double - expirationDate: String - gracePeriod: int - isGranted: boolean
+ CreditCard(cardID: int, clientName: String, issuerBank: String, bankAccount: String, balanceAmount: double, cvcNumber: int, interestRate: double, expirationDate: String) + getCvcNumber(): int + getCreditLimit(): double + getInterestRate(): double + getExpirationDate(): String + getGracePeriod(): int + getIsGranted(): boolean + setCreditLimit(newCreditLimit: double, newGracePeriod: int): void + cancelCreditCard(): void + display(): void

Figure 4 ; Credit Card Class

2.4. Inheritance Diagram

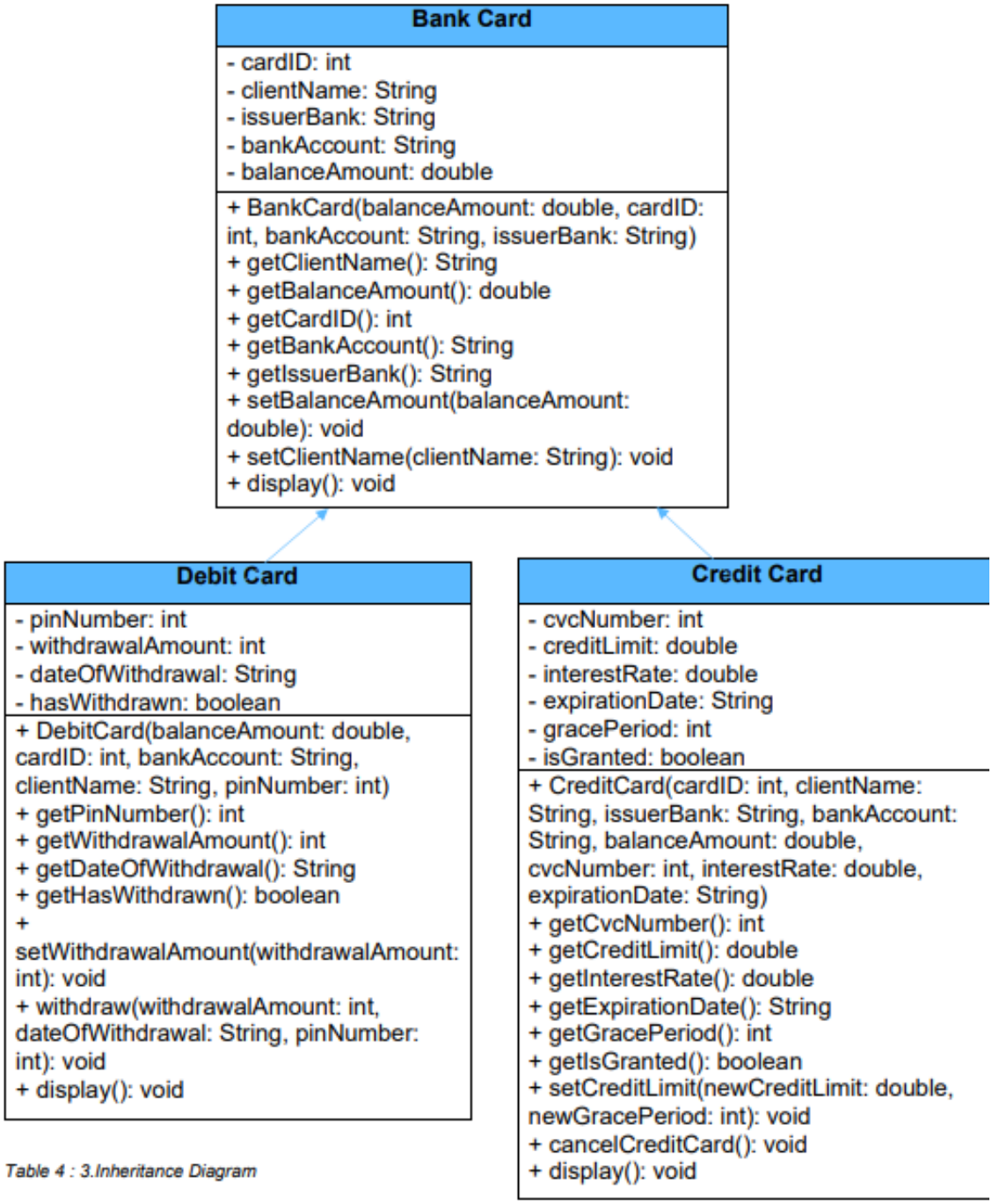

Table 4 : 3.Inheritance Diagram

Figure 5 ; Inheritance Class Diagram

2.5. BankGUI

Figure 6 ; Bank GUI Class Diagram

BANKGUI
- Frame : JFrame - cardIdLabel , clientNameLabel, issuerBankLabel, bankAccountLabel, balanceAmountLabel, cardIdLabel2, pinNumberLabel, withdrawalAmountLabel, withdrawalDateLabel, pinNumberLabel2, cvcNumberLabel, creditLimitLabel, interestRateLabel, gracePeriodLabel, expirationDateLabel, CcardIdLabel, CissuerBankLabel, CclientNameLabel, CbankAccountLabel, CbalanceAmountLabel, cardIdLabel4 : JLabel - cardId, clientName , issuerBank , bankAccount , balanceAmount , cardIdTextField2 , pinNumber , withdrawalAmount , pinNumberTextField2 , cvcNumber , creditLimit , interestRate , gracePeriod , CcardId , CissuerBank , CclientName , CbankAccount, CbalanceAmount, cardIdTextField4 : JTextField - displayBtn , clearBtn , addDebitCardButton , withdrawBtn , addCreditCardBtn, setCreditLimitBtn, cancelCreditCardBtn : JButton - dayOfWithdrawalComboBox , monthOfWithdrawalComboBox , yearOfWithdrawalComboBox , dayOfExpirationComboBox, monthOfExpirationComboBox, yearOfExpirationComboBox : JComboBox
+ (Constructor) BankGUI + actionPerformed(ActionEvent e) : void + static main (String args) : void

3. Pseudocode

IMPORT java.util.*;

IMPORT javax.swing.*;

IMPORT java.awt.event.*;

IMPORT java.awt.color.*;

CREATE BankGUI class that implements ActionListener

DECLARE JFrame.

DECLARE JLabel for Debit and Credit card.

DECLARE JTextField for Debit and Credit card.

DECLARE JButton for Debit and Credit card.

DECLARE JComboBox for all the components needed.

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

CREATE a constructor BankGUI()

DO

INITIALIZE ArrayList <BankCard> cards

INITIALIZE JFrame.

INITIALIZE JLabels.

INITIALIZE JTextFields.

INITIALIZE JButtons.

SET Bounds for the JLabels

SET Bounds for the JTextFields

SET Bounds for the JButtons

INITIALIZE JLabel for Debit card.

INITIALIZE JTextfield for Debit card.

INITIALIZE JComboBox for Debit card.

INITIALIZE JButtons for Debit card.

SET bounds for debit card JLabels

SET bounds for debit card JTextFields

SET bounds for debit card JButtons

SET bounds for JComboBox

INITIALIZE Credit card JLabels

INITIALIZE Credit card JTextFields

INITIALIZE Credit card JButtons

INITIALIZE Credit card JComboBox

SET bounds for Credit card JLabels

SET bounds for Credit card JTextFields

SET bounds for Credit card JButtons

SET bounds for Credit card JComboBox

ADD components to frame for Debit card.

ADD components to frame for Credit card.

SET size for the frame, setResizable (false), setLayout (null), setVisible (true)

ADD action listener to addDebitCardButton

ADD action listener to addCreditCardButton

ADD action listener to withdrawBtn

ADD action listener to setCreditLimitBtn

ADD action listener to cancelCreditCardBtn

ADD action listener to displaybtn

ADD action listener to clearBtn

END DO

CREATE actionPerformed with parameter e as ActionEvent

DO

if e.getSource() == addDecitCardButton

DO

cardId.getText(); OR
clientName.getText(); OR
issuerBank.getText(); OR
bankAccount.getText(); OR
balanceAmount.getText(); OR
pinNumber.getText(); OR

DO

SHOW error in Dialog Box

END DO

ELSE

TRY

DO

getTexts for cardId, clientName, issuerBank, bankAmount balanceAmount, pinNumber put them in variables

debitCardExists equals false

If bankCardArray is empty

DO

debitcard = new DebitCard (cardId, bankAccount, balanceAmount, issuerBank, clientName, pinNumber);

ADD debitCard obljject to bankCardArray

SHOW success message in Dialog box

END DO

ELSE

DO

```
                    FOR BankCard card : bankCardArray
                     If bankcard instanceof DebitCard
                DO
                    newDebitCard = (DebitCard) card
                    if  newDebitCard.getcardId == cardId
                DO
                    debitCardIdExists = true
                SHOW debit card exists error in dialog box
                    BREAK
                END DO
            END IF
        END FOR
    If
        DO
```

Debitcard = new DebitCard (balanceAmout, cardId, bankAccout, issuerBank, ClientName, pinNumber)

ADD debitCard object to BankCardArray

SHOW added successfully message in dialogbox

END DO

END IF

CATCH (NumberFormatException ex)

DO

SHOW error in DialogBox

END DO

END IF

DO

If (e.getSource() == addCreditBtn)

DO

CcardId.getText(); OR

CclientName.getText(); OR

CissuerBank.getText(); OR

CbankAccount.getText(); OR

CbalanceAmounr.getText(); OR

DO

SHOW error id dialogBox

END DO

ELSE

TRY

DO

CreditCardExists = false

If BankCardArray is empty

DO

Gettexts for cardID, clientName, issuerBank, bankAccount, balanceAmount, cvcNumber, interestRate, pinNumber, expirationDate and put them in variables.

debitCardExists = false

if BankCardArray is empty

DO

CreditCard = new creditCard

(cardId, clientName, issuerBank, bankAccount, balanceAmount, cvcNumber, interestRate, expirationDate, pinNumber)

ADD credit object to BankCardArray

SHOW success message in dialog box

END DO

ELSE

DO

FOR BankCard card : cards

If card instanceof CreditCard

DO

newCreditCard = (CreditCard) cards

if newCreditCard.getCardID() == cardId

DO

creditCard exists equals true

SHOW debitCard exists error in dialogBox

BREAK

END DO

END IF

END FOR

DO

creditCard = new Creditcard

(cardId, clientName, issuerBank, bankAccount, balanceAmount, cvcNumber, interestRate, expirationDate, pinNumber)

ADD creditCard object to BankCardArray

SHOW success message in dialogBox

END DO

END IF

CATCH (NumberFormatException ex)

DO

SHOW error in dialogBox

END DO

END IF

DO

If(e.getSource() == withdrawBtn)

DO

getTexts for cardId, withdrawAmount, balanceAmount, pinNumber, dates and put them in varirables.

if withdrawn is false

if cardIdStr is empty OR withdrawalAmountStr is empty or pinNumberStr is empty.

DO

SHOW error message in dialogBox

END DO

ELSE

DO

cardId, withdrawnAmount, pinNumber, balanceAmount as integer

TRY

cardIdStr, withdrawalAmountStr, pinNumberStr, debitCard balanceAmount change from string to integer and put in varirables.

If BankCard is empty

DO

SHOW error in dialogBox

END DO

ELSE

DO

FOR BankCard card : cards

If (BankCard instanceof DebitCard)

DO

debitCard equals (DebitCard) BankCard

if debitCard.getcardId() == cardId

DO

debitCard present is true

if debitCard.getpinNumber() == pinNumber

DO

If withdrawalAmount less than balanceAmount

DO

CALL withdraw method

SHOW success message

If withdrawn is true

END DO

ELSE

DO

SHOW insufficient balance

END DO

Else

DO

SHOW pin does not match

```
                    END DO

                    END IF

                    BREAK

END IF

END FOR

            If debitCard not present

            DO

                    SHOW error in dialog box

            END DO

            END IF
END IF

CATCH (NumberFormatException e)

        DO

            SHOW error in dialogBox

        END DO

END
```

```
        END IF
END DO
        END IF
END DO
        DO
        If(e.getSource() equals setCreditLimitBtn)
        DO
        if cardId is empty OR
        if creditLimit is empty OR
        if gracePeriod is empty
                DO
                        SHOW error message in dialogBox
                END DO
        ELSE
                TRY
```

Get values for cardId, creditLimit, gracePeriod and put them in variables.

FOR BankCard credit : BankCardArray

If (credit card instanceof CreditCard)

DO

ggCard equals (CreditCard) credit

If ggCard.getCardID() == cardId

DO

CALL setCreditLimit (creditLimit, gracePeriod) methods from ggcard object.

cardIdFound is true

BREAK

END DO

END IF

END DO

END FOR

```
IF cardIdFound
        DO
            SHOW success message in dialogBox
        END DO
    ELSE
        DO
            SHOW error message in dialogBox
        END DO
    END IF
CATCH (NumberFormatException ex)
    DO
        SHOW error message in dialogBox
    END DO
END
    END IF
```

END DO

DO

If (e.getSource() == cancelCreditCardBtn)

DO

If all text fields are empty

DO

SHOW error in dialogBOx

END DO

ELSE

TRY

Get cardId, creditLimit, gracePeriod values and put in variables

CardId equal is false

FOR BankCard credit : BankCardArray

DO

If credit instanceof CreditCard

DO

```
                    Creditobject equals (CreditCard) Credit
                        If creditobject.getcardId() == cardID

                    DO

                        CALL cancelCreditCard() of creditobject

                        cardIdEqual is true

                    BREAK

                END DO

            END DO

        END IF

    END FOR

    IF iscardIDEqual

                    DO

                        SHOW successful message in dialogBox

                    END DO

                ELSE
```

DO

SHOW error message in dialogBox

END DO

END IF

CATCH (NumberFormatException ex)

DO

SHOW error in dialogBox

END DO

END

END IF

DO

If (e.getSource() equals displayBtn

DO

Get values of cardId, clientName, issuerBank, bankAccount, balanceAmount, pinNumber, withdrawalAmount, dates and store in variable.

IF BankCardArray is empty

DO

```
            SHOW  error message in dialogBox

        END DO

    ELSE
        DO

            FOR BankCard card : BankCardArray

                If(card instanceof DebitCard)

        DO

            debitCard equals (DebitCard) card

            CALL display method

            Display details stored in displayPopUp

            SHOW displayPopUp in dialog box

        END DO

    ELSE

        DO

            SHOW error message in dialogBox
```

```
            END DO
        END IF
END FOR
            END DO
        END IF
END DO
        END IF
END DO
            DO
                If ( e.getSource() equals clearBtn)
                DO
                    SET cardID to “ ”
                    SET CcardId to “ “
                    SET clientName to “ ”
                    SET CclientName to “ “
```

```
            SET issuerBank to “ ”

            SET CissuerBanK to “ “

            SET bankAccount to “ “

            SET CbankAccount to “ “

            SET balanceAmount to “ “

            SET CbalanceAmount to “ “

            SET pinNumber to “ “

            SET CpinNumber to “ “

            SET withdrawalAmount to “ “

            SET cvcNumber to “ “

            SET creditLimit to “ “

            SET interestRate to “ “

            SET gracePeriod to “ “

        END DO

    END IF
```

END DO

DO

If (e.getSource() equals displayBtn)

DO

Get values of cardId, issuerBank, clientName, bankAccount, balanceAmount, cvcNumber, creditLimit, gracePeriod, interestRate, expirationDate put in variables.

IF BankCardArray is empty

DO

SHOW error in dialogBox

END DO

ELSE

DO

FOR BankCard card : cards

IF card instanceof CreditCard

DO

```
                    CreditCard equals (CreditCard) card

                    CALL display method of CreditCard

                    Display details stored in PopUp

                    SHOW PopUp in dialogBox

                    END DO

                ELSE

                    DO

                        SHOW error message in Dialogbox

            END DO

        END IF

    END FOR

  END DO

END IF

    END DO

        END IF
```

```
END DO
```

4. Method Description

Method Name	Method Description
actionPerformed(ActionEvent e)	The actionPerformed method, which incorporates the functionality of the buttons in the GUI and executes certain tasks when an event happens, must be used since the BankGUI class implements ActionListener.
addDreditCardBtn()	This button adds a debit card to the array list of the bank card class by accepting the values balance amount, cardID, bank account, issuer bank, clientName, and pinNumber.If the criteria of the input fields is not met, errors are consequently displayed.
addCreditCardBtn()	This button adds a credit card to the array list of the bank card class by taking the following values: cardID, clientName, issuerBank, bankAccount, balanceAmount, cvcNumber, interestRate, and expirationDate. If the requirements of the input fields are not met, errors are therefore displayed.

withdrawDebitCardBtn()	To withdraw money from an existing debit card, use this button. It accepts the parameters cardId, withdrawalAmount, withdrawalDate, and pinNumber. If the debit card does not exist or if the balance is insufficient, an error message will be displayed.
setCreditLimitBtn()	This button sets the creditLimit by using the cardId, creditLimit, and gracePeriod values. If specific conditions are met, the class of credit card with the stated credit limit is also referred to. If incorrect inputs are given, the proper error message will be displayed.
cancelCreditCardBtn	When the cancel button is pressed, the credit card is cancelled and the cancel credit method from the credit card class is called. The cancel button accepts the cardId as input. If the card id doesn't match, an error message will appear.
Clear Buttons	Clear buttons are used to easily erase the old data from text fields inside the GUI. As a result, the user can easily add new cards after removing the previous data using the clear button.
DisplayButtons	The relevant values supplied in the text fields are shown in a pop-up message usi buttons on both debit and credit card. Additionally, this calls the Display me previous class. Error is displayed when unusual activities are taken.

.

Table 1 ; Method of Description

5. Testing

5.1. Test 1

Test Number	1
Objective	Verify whether the program can be compiled and run through command prompt, including the screen shot.
Action	-Finding the program path and open command prompt -Enter the commands a) Java BankGUI.Java b) Javac- Xlint unchecked
Expected Result	The program should be running through command prompt.
Actual Result	The program was compiled and was running smoothly.
Conclusion	The test was successful.

Table 2 : Test 1

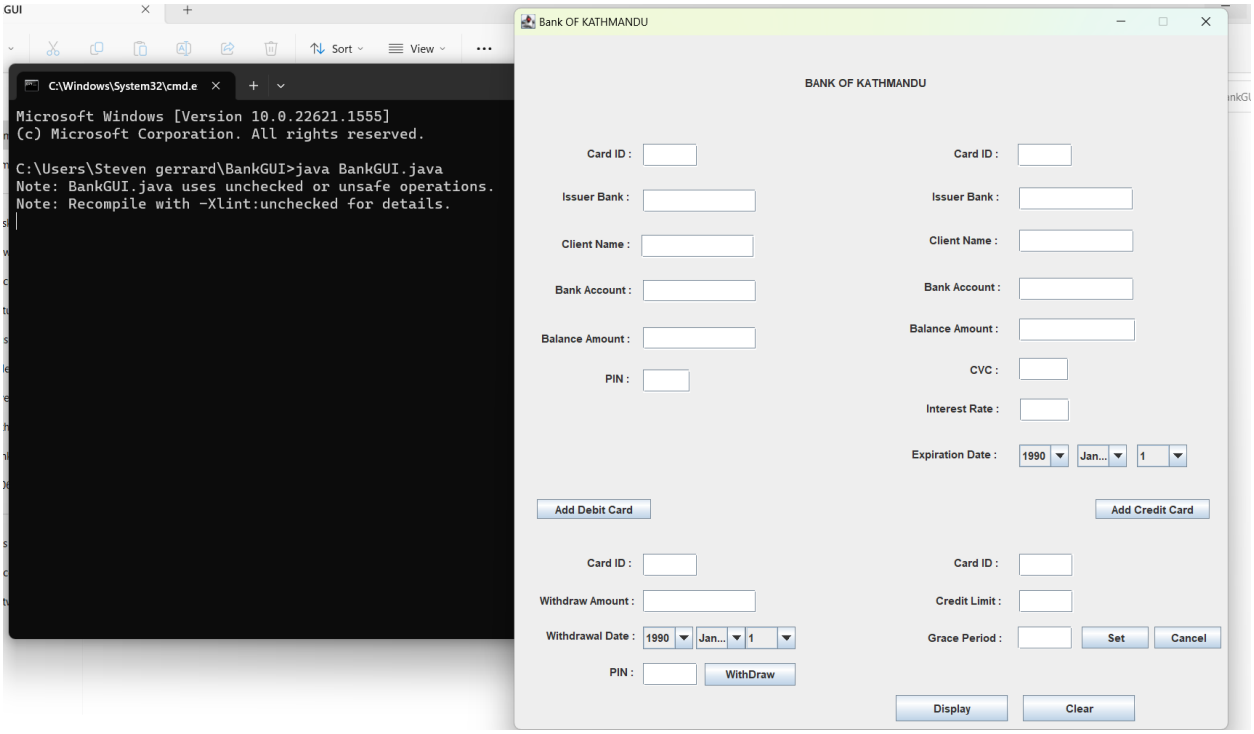

Figure 7 : Screenshot of command prompt

5.2. Test 2

a) Add debit card

Test Number	2
Objective	Add debit card.
Action	Following inputs were taken: Card Id : 22 Client Name : Samir Gurung Issuer Bank: Kumari Bank Bank Account : ahai22 BalanceAmount : 80000 PIN number : 9812
Expected Result	The debit card would have been added.
Actual Result	The debit card was added.
Conclusion	The test was successful.

Table 3 ; Add debit card'

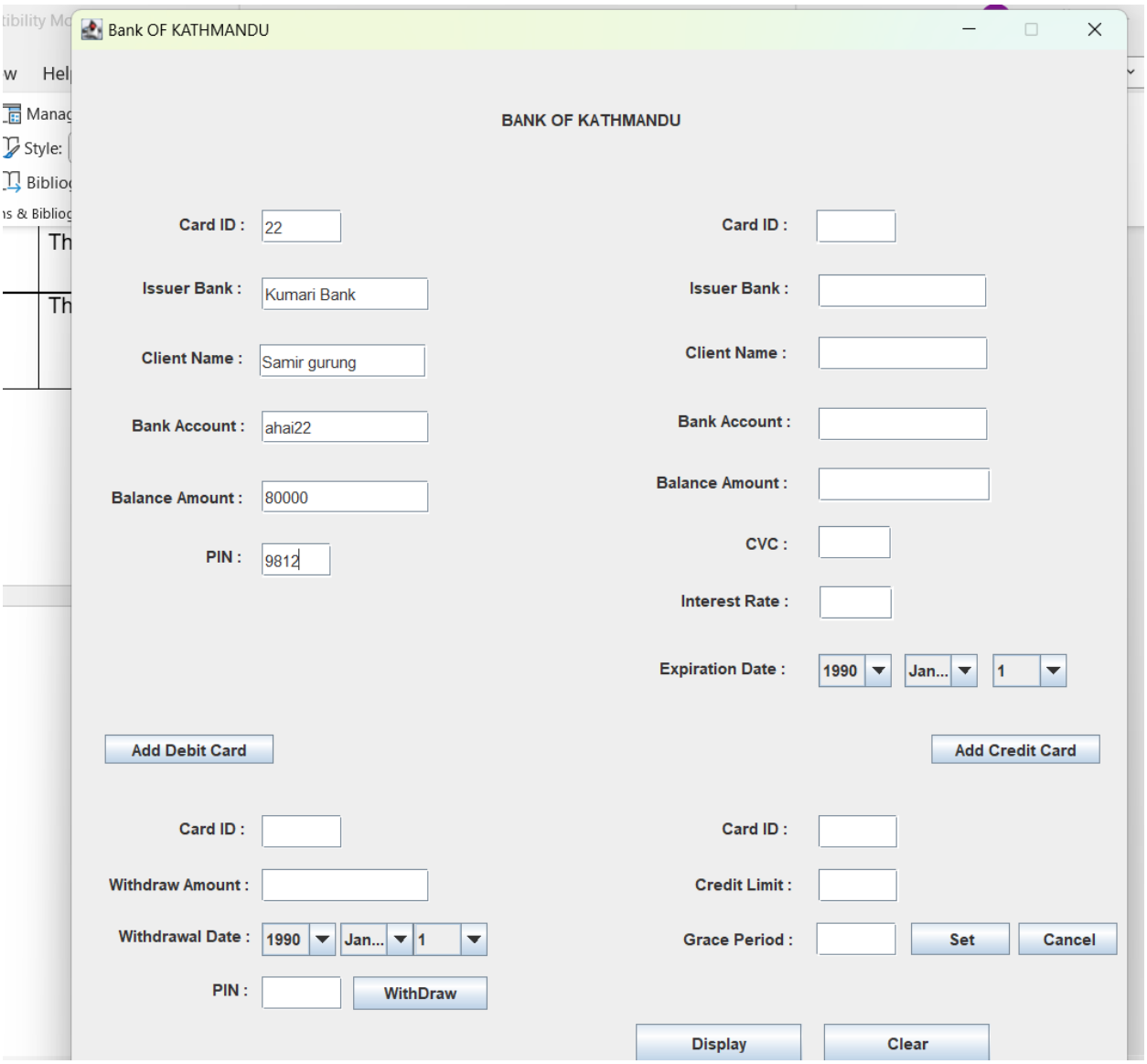

Figure 8 :Add Debit Card

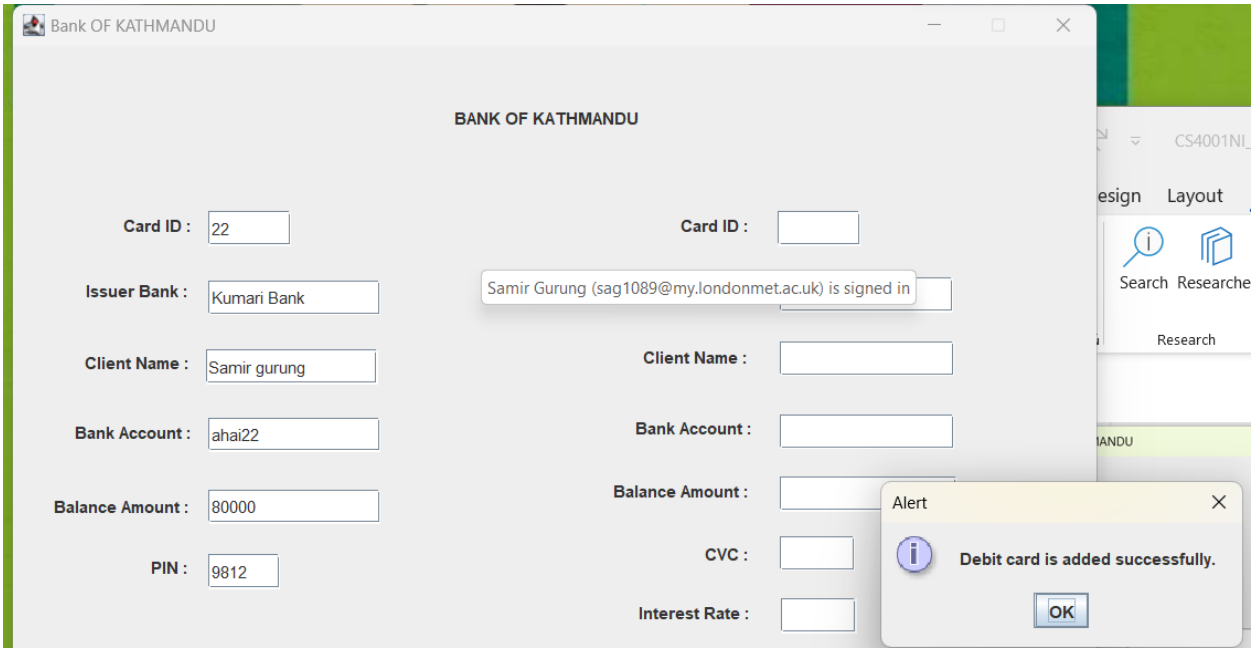

Figure 9 : Screenshot of Successfully added Debit Card

b) **Add credit card**

Test Number	2
Objective	Add credit card.
Action	Following inputs were taken: Card Id : 22 Client Name : Samir Gurung Issuer Bank: Kumari Bank Bank Account : ahai22 BalanceAmount : 80000 CVC number : 1222 Interest Rate : 12 Expiration Date: 1 Jan 2016
Expected Result	The credit card would have been added.
Actual Result	The credit card was added.
Conclusion	The test was successful.

Table 4 : Add credit card

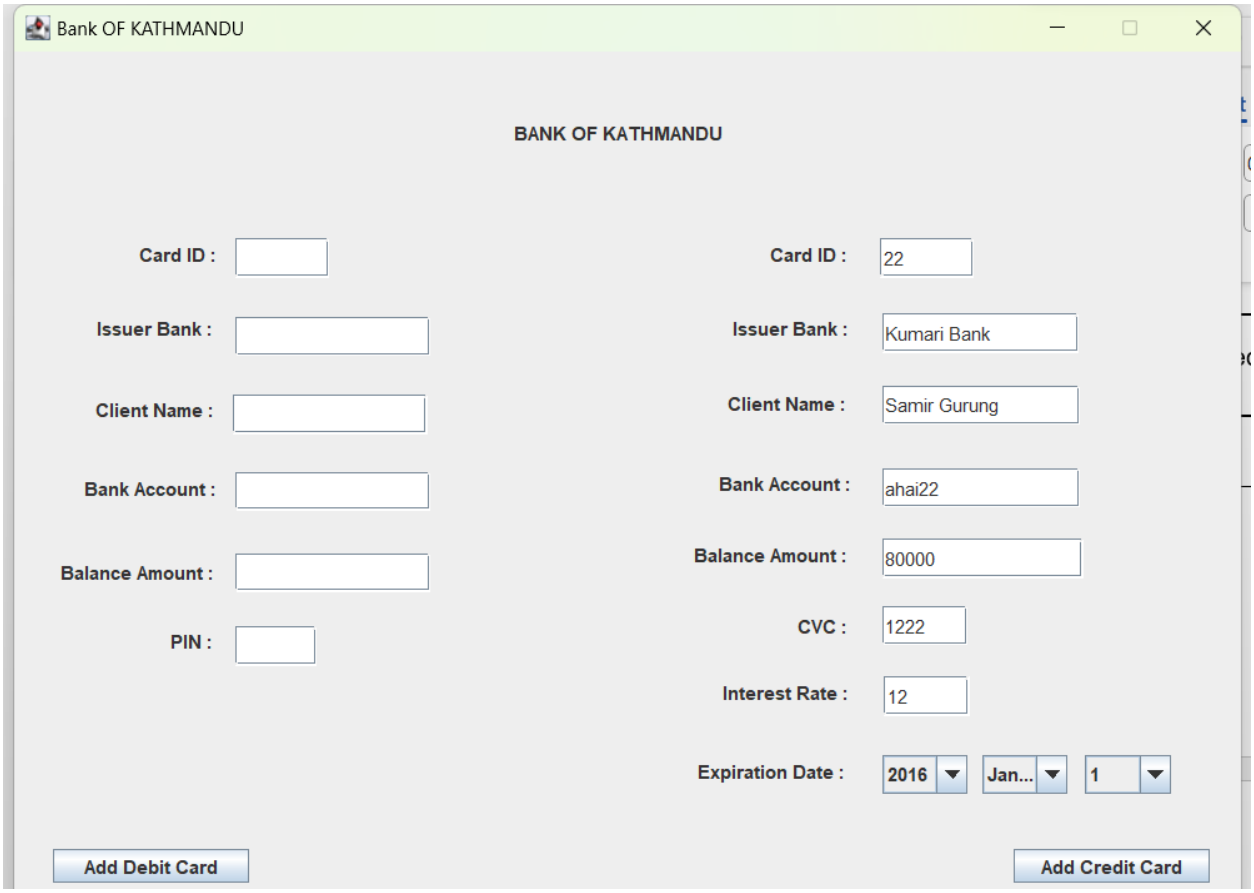

Figure 10 : Credit Card Added

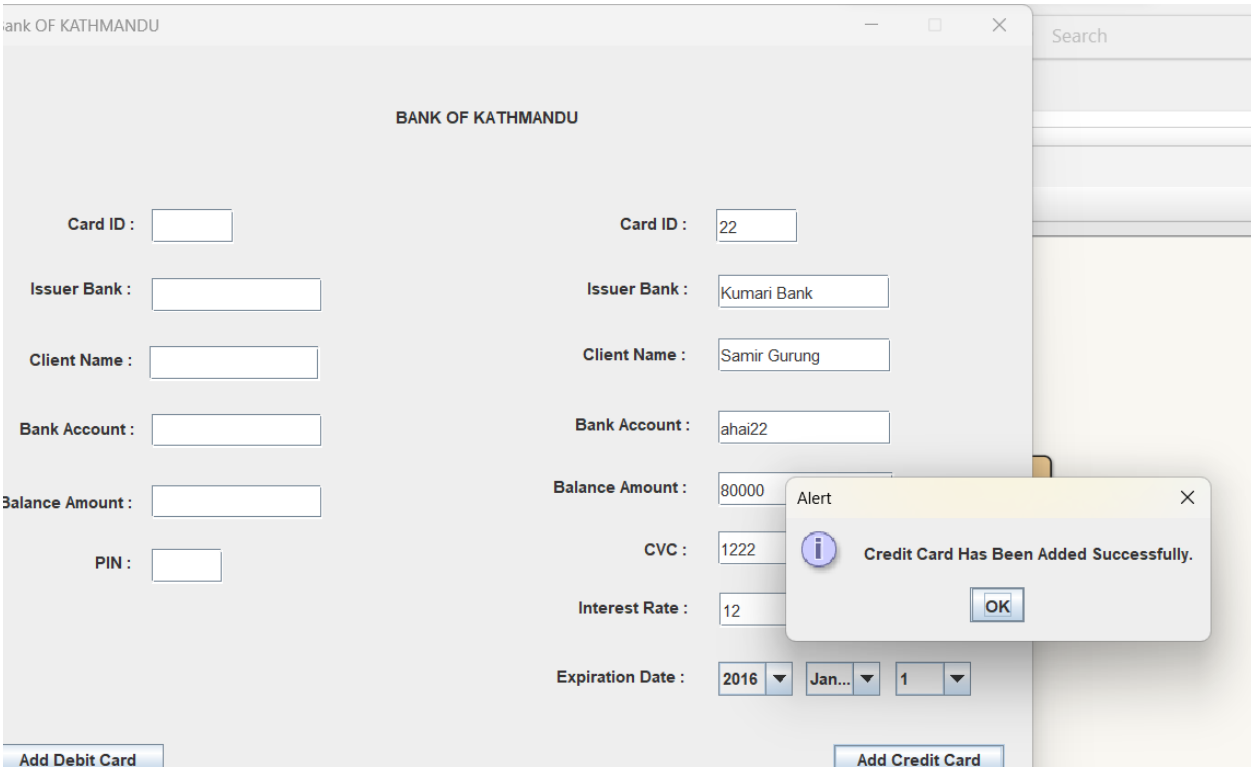

Figure 11: Successfully added Credit Card

c) **Withdraw from debit card**

Test Number	2
Objective	Withdraw amount from debit card.
Action	Following inputs were taken: Card Id: 11 Withdrawal Amount: 100 Date of Withdrawal: 1990/Jan/1 Pin number : 11
Expected Result	The amount should have been withdrawn.
Actual Result	The amount was withdrawn
Conclusion	The test was successful.

Table 5 :Test Withdraw

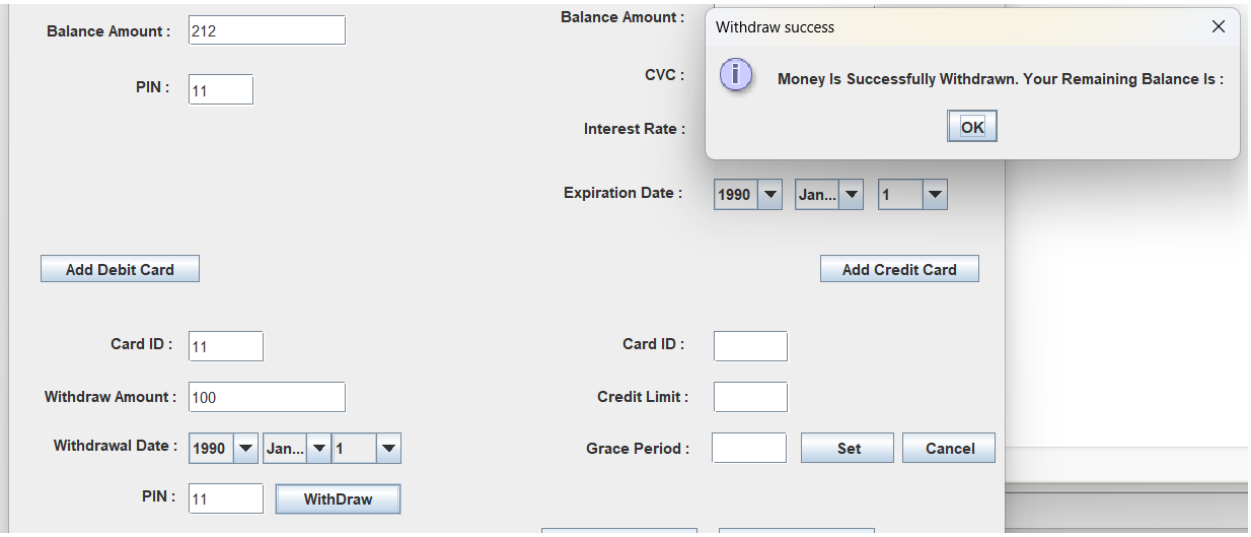

Figure 12 : Money Withdrawn Successfully

d) **Set Credit Limit**

Test Number	2
Objective	Set the credit limit.
Action	Following inputs were taken: Credit Limit :122 Grace Period :12
Expected Result	The credit limit would have been set.
Actual Result	The credit limit was set.
Conclusion	The test was successful.

Table 6 : Test Set Credit limit

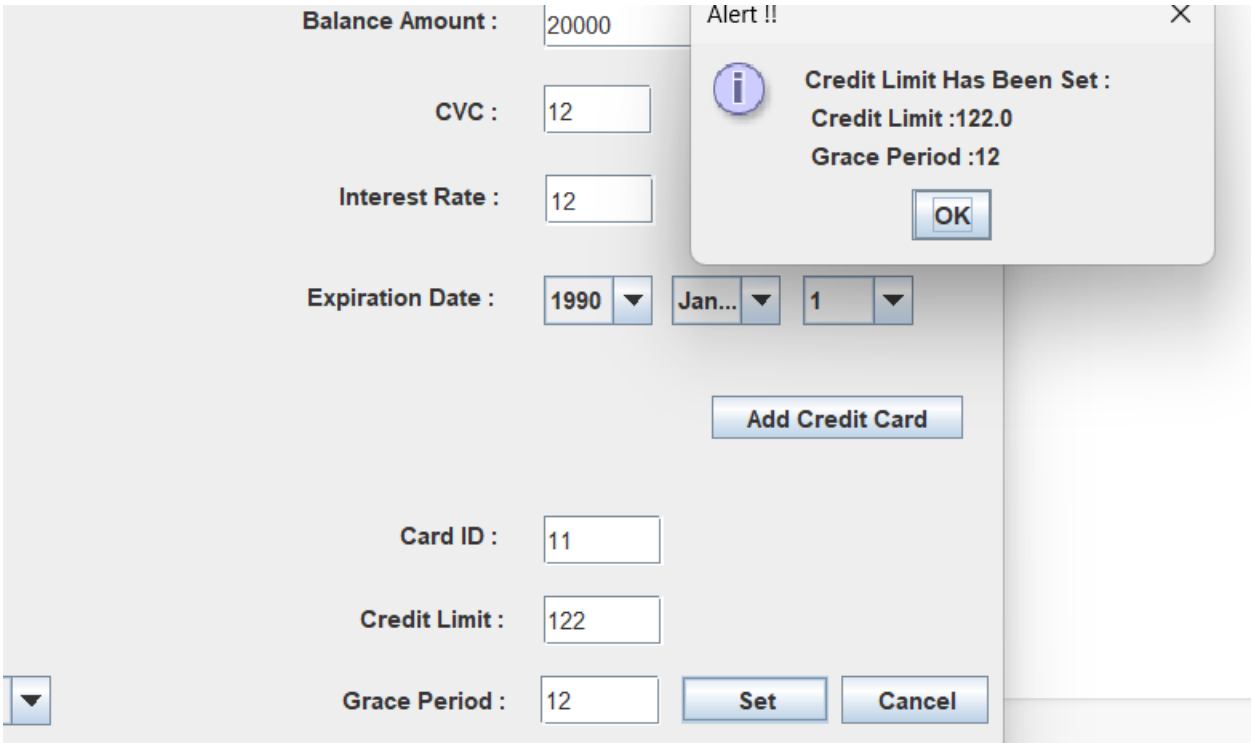

Figure 13 : Set Credit Card Limit

c) Remove the credit card

Test Number	2
Objective	Cancel the credit card.
Action	Credit card was added then card Id was taken to cancel the card.
Expected Result	The credit card would have been cancelled.
Actual Result	The credit card was cancelled.
Conclusion	The test was successful.

Table 7 : Cancel Credit Card

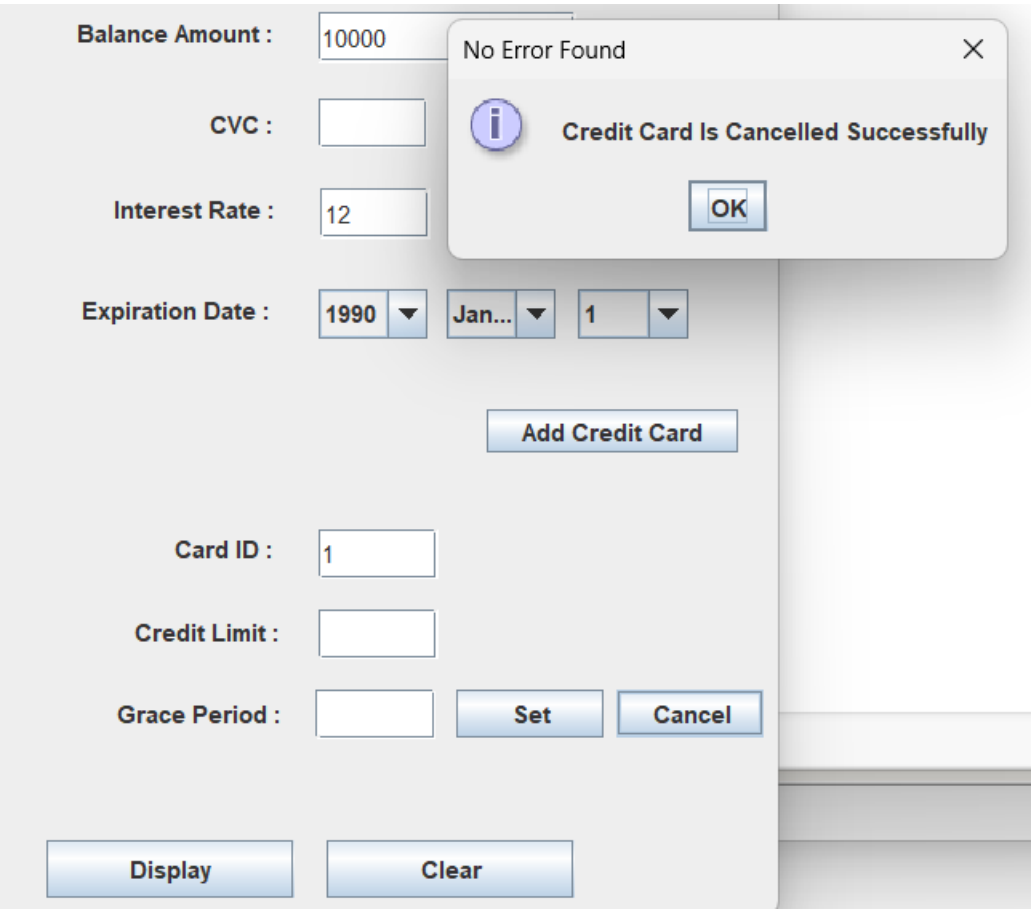

Figure 14 : Credit Card Cancel Success

5.3. Test 3

Test Number	3.1
Objective	Show popup when unsuitable values were input for card Id
Action	String value was added instead of int.
Expected Result	Suitable pop up message would have been appeared.
Actual Result	Suitable pop up appeared.
Conclusion	The test was successful.

Table 8 : Test 3.1

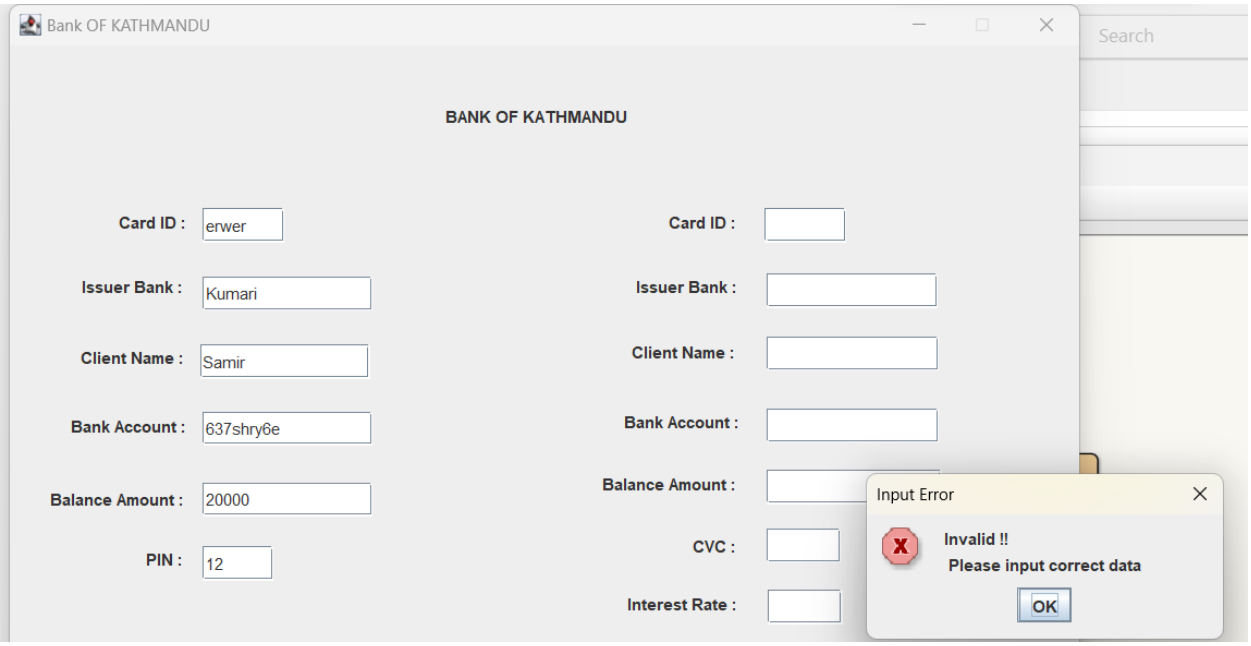

Figure 15 : Invalid Input detected

Test Number	3.2
Objective	Show popup when same debit card was added twice.
Action	Same debit card was added twice
Expected Result	Suitable pop up message would have been appeared.
Actual Result	Suitable pop up appeared.
Conclusion	The test was successful.

Table 9 : Test 3.2

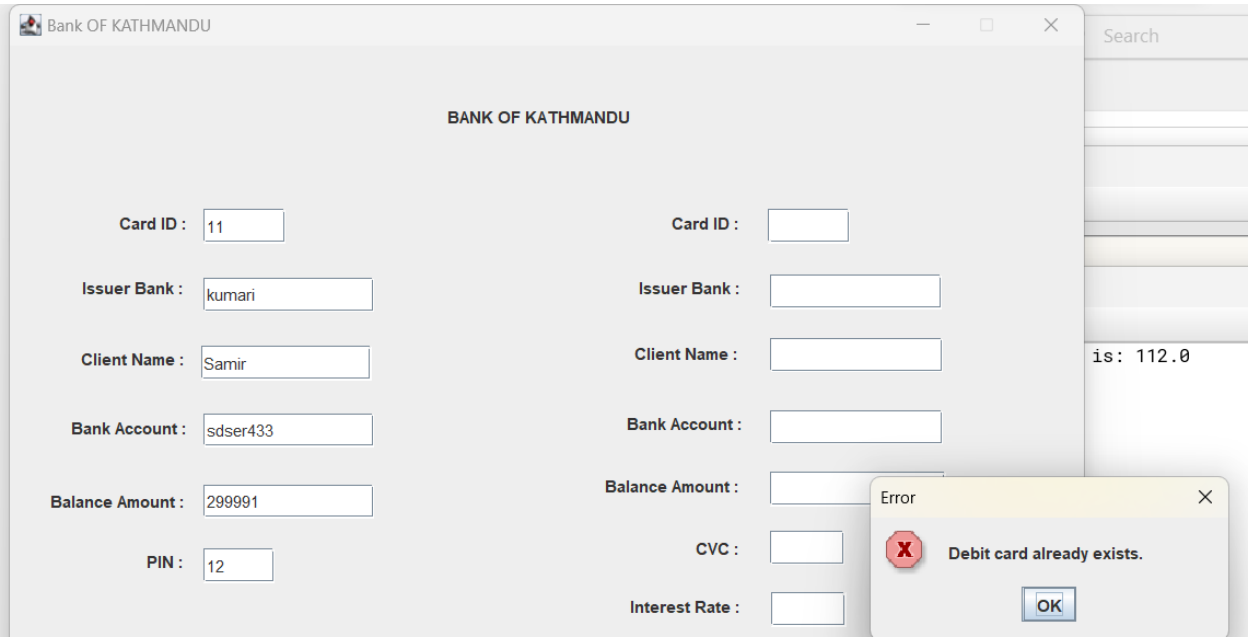

Figure 16 : Debit card already exists

6. Error detection and correction

6.1. Syntax error

An error in syntax occurs when a programmer uses the syntax of a coding or programming language incorrectly. A compiler is a piece of software that checks programs for syntax errors that must be addressed by the programmer before the program is executed.

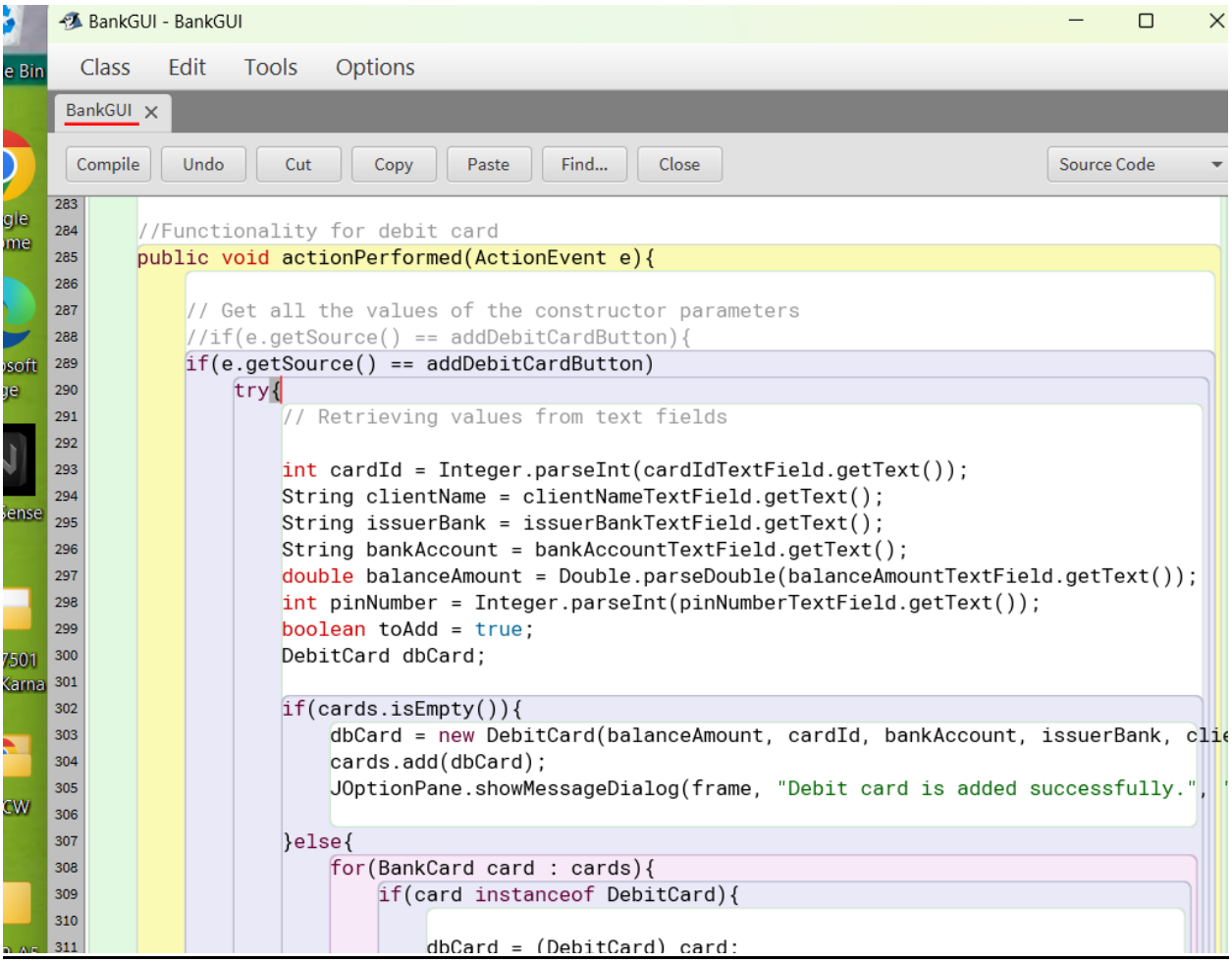

Figure 17 : Syntax Error

An “{“ was missing so the error was detected while compiling, which is an Syntax error.

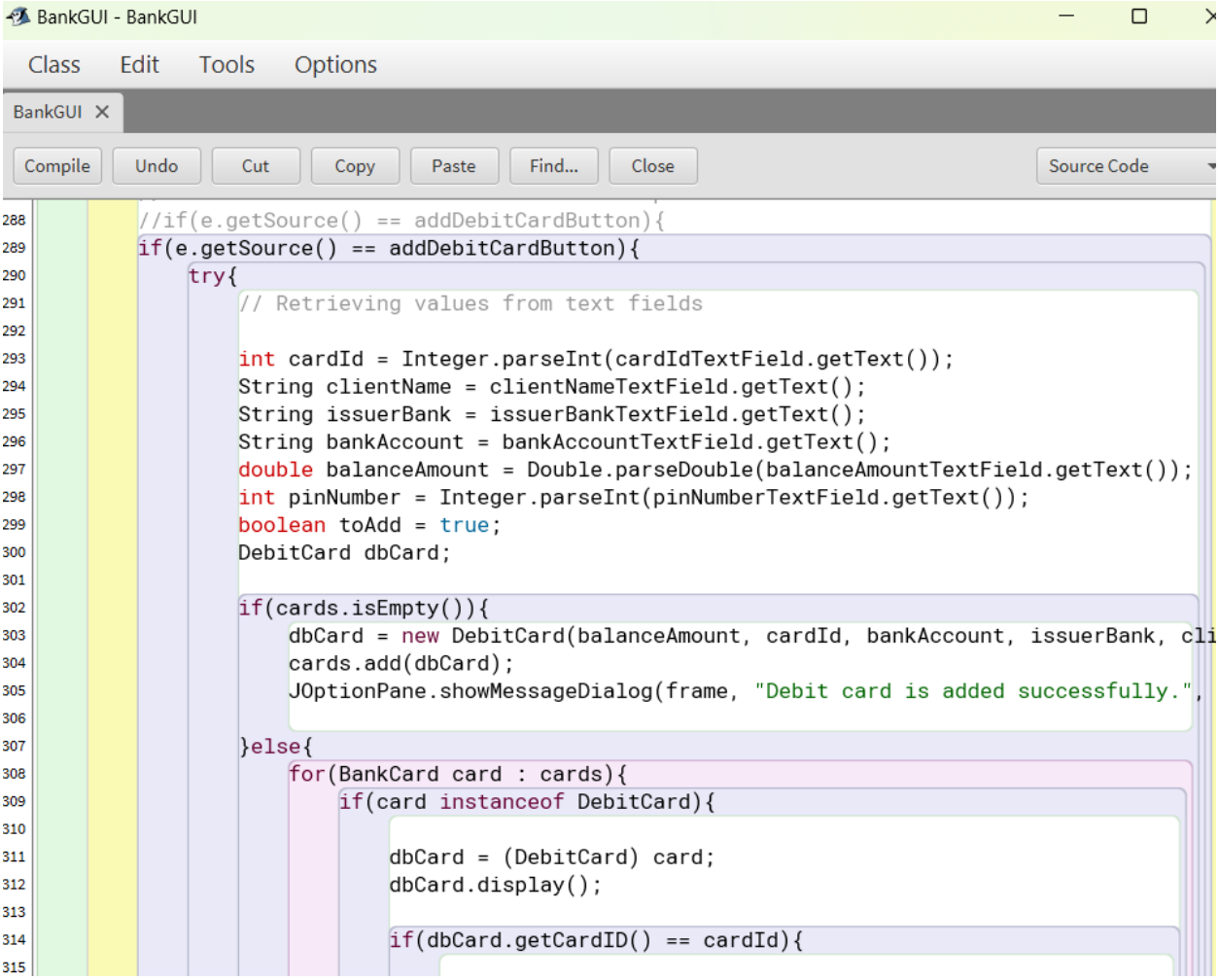

Figure 18 : Correction of Syntax Error

6.2. Semantic error

An error which is grammatically correct but has no logic is Semantic error.

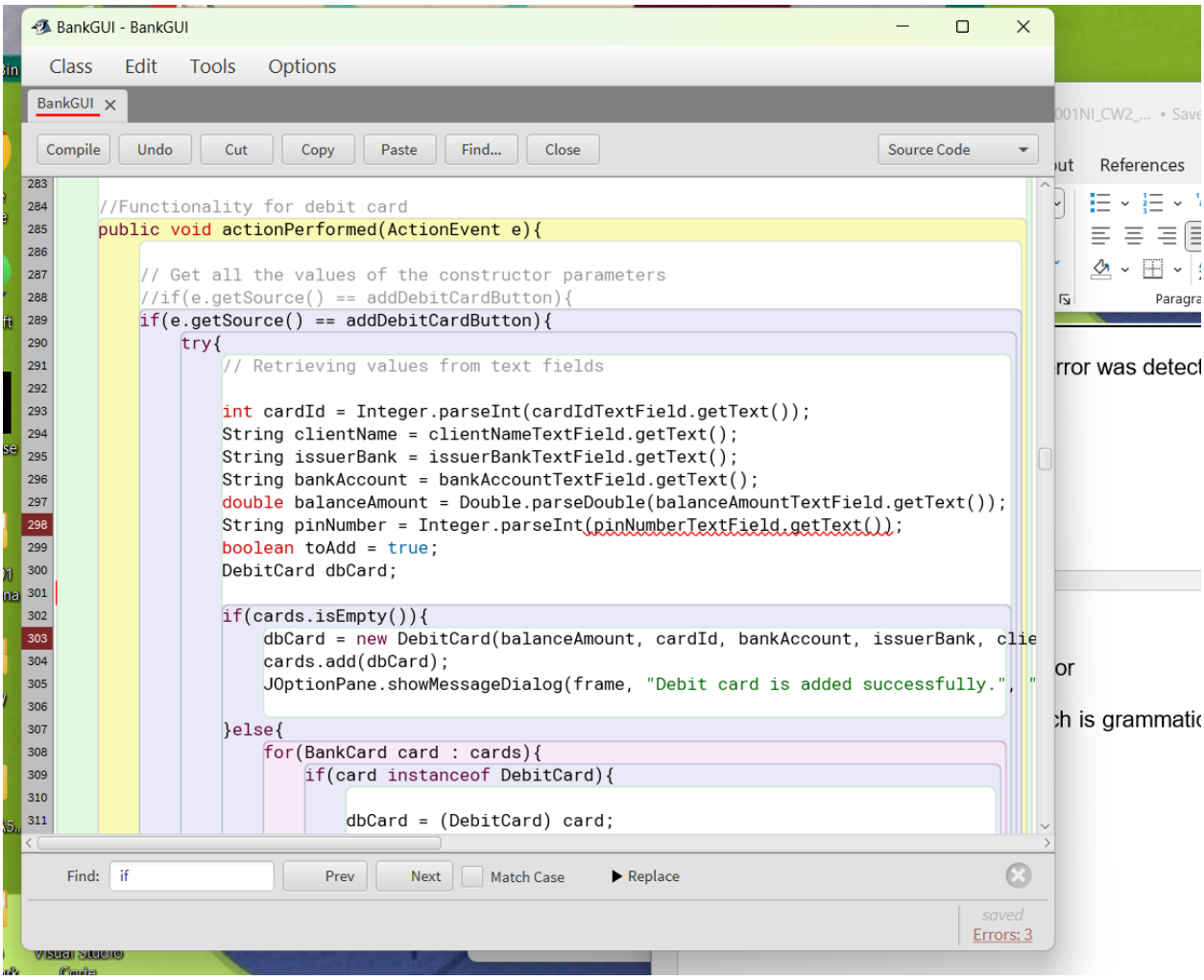

Figure 19 : Semantic Error

There was an error on data type where “String” should be corrected with “int”

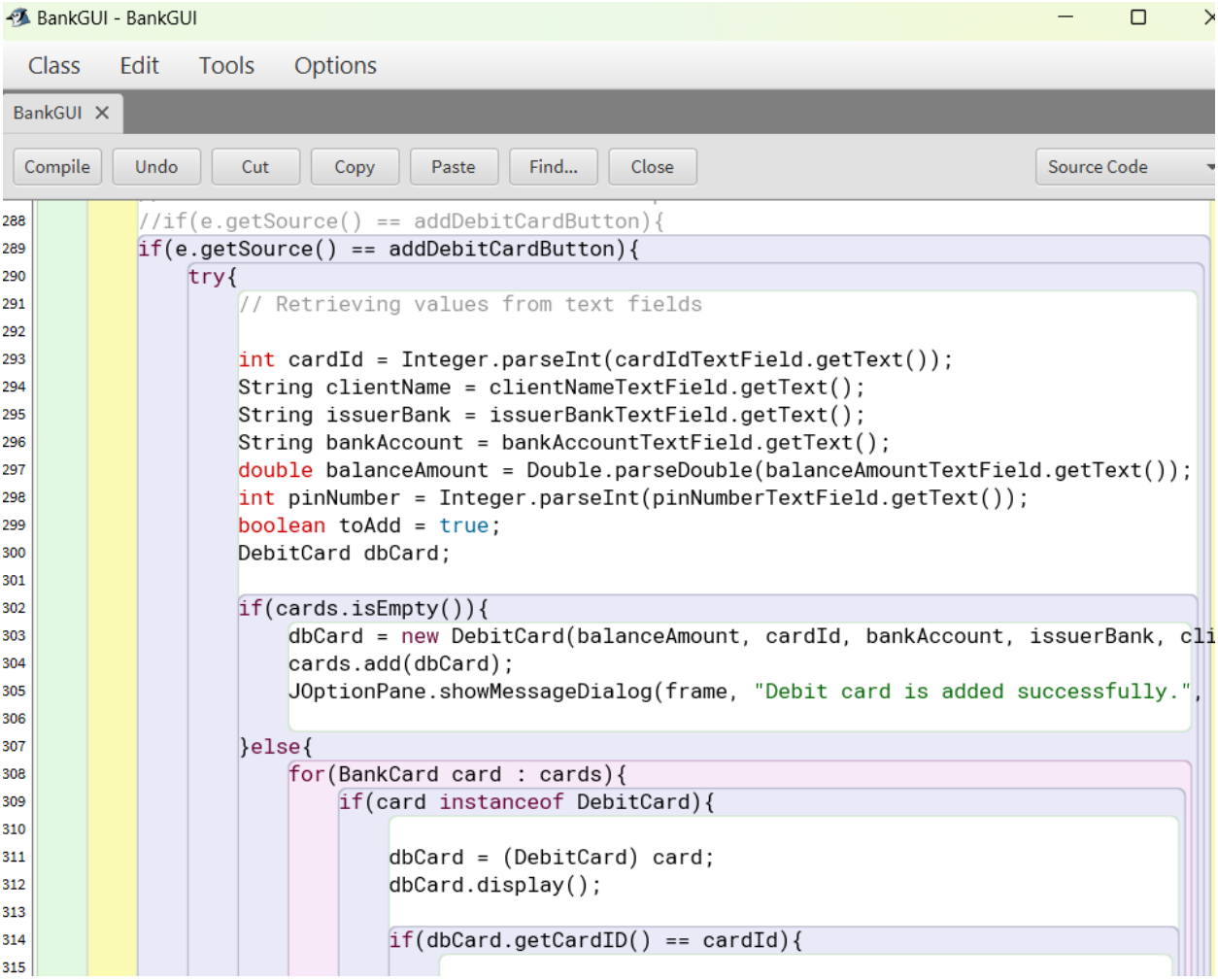

Figure 20 : Correction Of Semantic Error

6.3. Logical error

An Logical error was detected because there was double “==” instead of one.

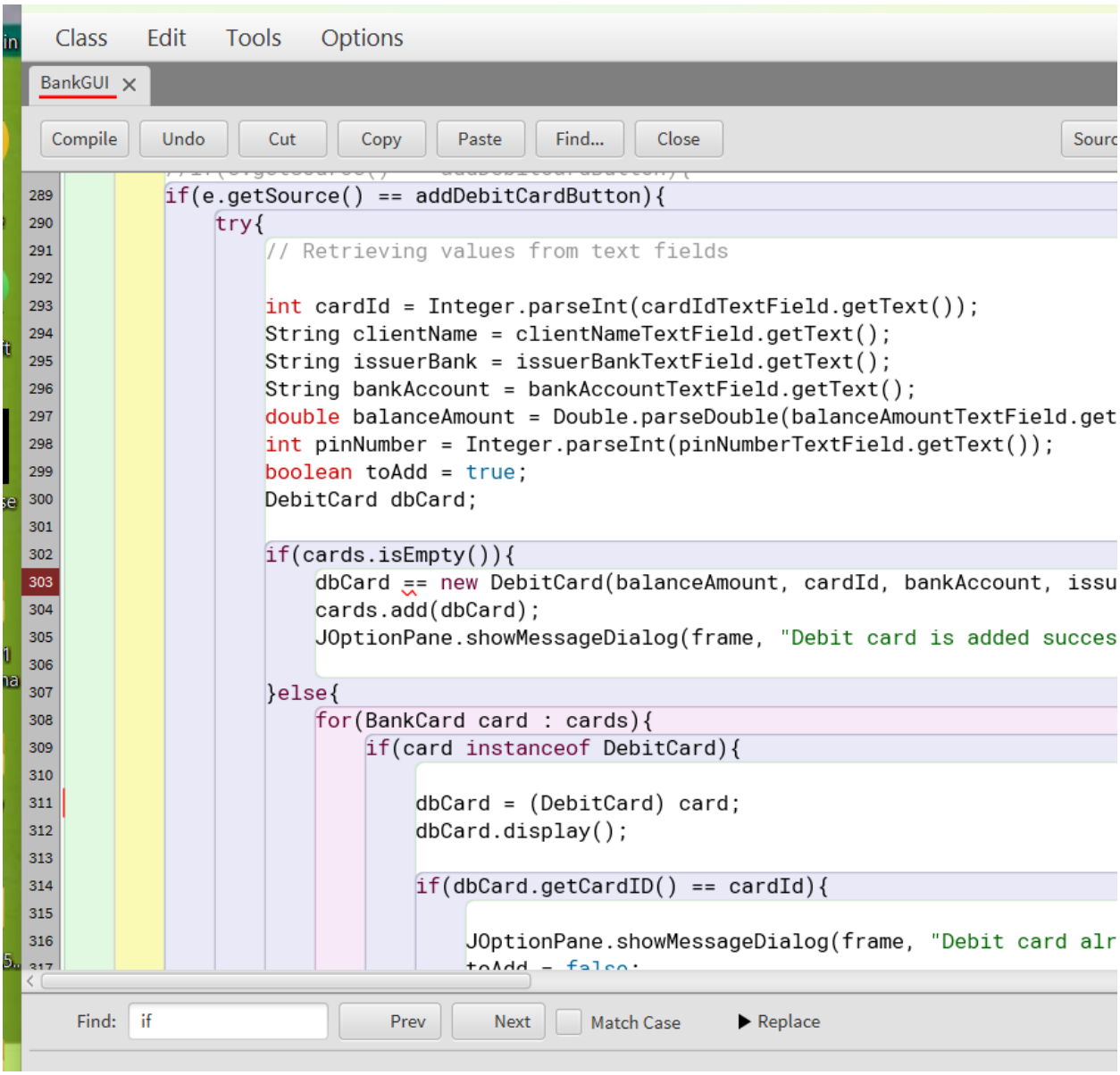

Figure 21 : Logical Error

The error was fixed and the program compiled successfully.

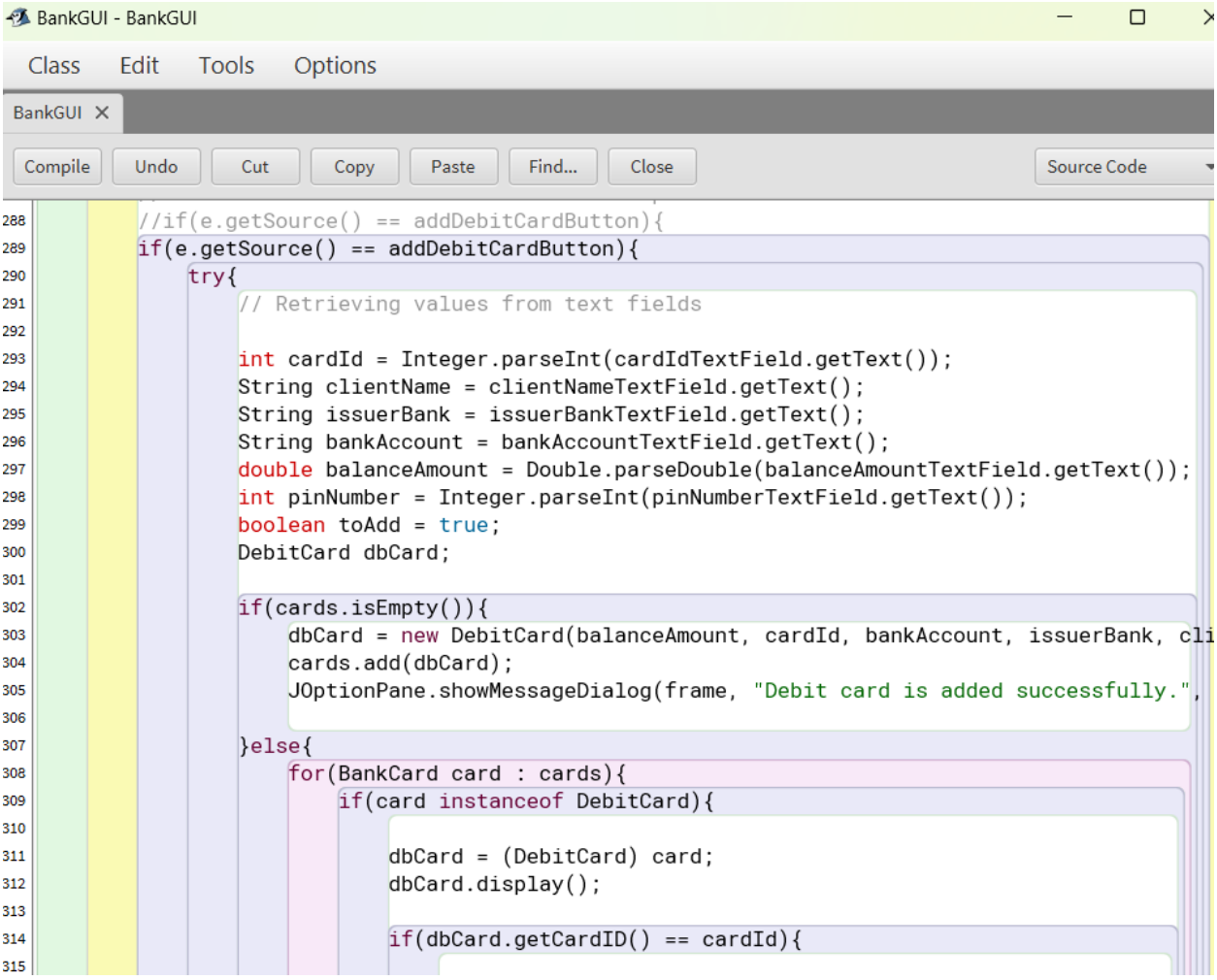

Figure 22 : Correction Of Logical Error

7. Conclusion

Adding a GUI for a system that stores details of Bank card in an arraylist has really been an exciting and complicating part of the coursework for me. It was exciting because making a GUI based programs was new to me and it was related to our real life scenario which was bank cards. And it was complicating because it was my first time working on GUI based programs, but with lots of research and trying many times on coding and all I was able to get through it. It was also possible with the help and guidelines from my teachers and friends that I was able to complete my course work. I have to say that I have learned more java components and got better on programming a GUI based programs.

The course work has provided us an opportunity to design and develop a graphical user interface of a bank card which was kind of new to us. Through this course work we got to know more java components and its uses.The making of BankGUI required combination of both programming and technical knowledge which was not that easy. We had to construct a functional GUI for a bank cards of debit card and credit card where we can use it to add a debit card and a credit card, withdraw money from debit card, set a credit limit, or cancel the credit card of the costumer. We can see all the information of a costumer on text fields or clear it through display and clear button. A suitable message will be shown if a user or a costumer enters an invalid information. Overall the course work gave us an opportunity to try our skills and knowledge on GUI designing and programming for a real world scenario.

References

Techopedia (2013, October 10) . Introduction of bluej. Retrieved from

https://www.lifewire.com/microsoft-word-4159373

Ballew, J. (2023, January 25). What Is Microsoft Word? Retrieve from

https://www.lifewire.com/microsoft-word-4159373

Pedamkar, P. (2023, January 25). Introduction to Class Diagram. Retrieved from

https://www.educba.com/class-diagram/

Appendix

- **BankCard**

```
/**
 * Write a description of class Bankcard here.
 *
 * @author (22015816 Samir Gurung)
 * @version (1.0.0)
 */
public class BankCard
{
    //attributes for BankCard class
    private int cardID;
    private String clientName;
    private String issuerBank;
    private String bankAccount;
    private double balanceAmount;

    //constructor
    public BankCard(double balanceAmount,int cardID, String bankAccount, String issuerBank)
    {
```

```
        //initializes client name to an empty string
        this.clientName = "";
        //assigns attributes with the parameter values
        this.balanceAmount = balanceAmount;
        this.cardID = cardID;
        this.bankAccount = bankAccount;
        this.issuerBank = issuerBank;
    }

    //getter or accessor methods for all the attributes
    public String getClientName()
    {
        return this.clientName;
    }

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

    public int getCardID()
    {
        return this.cardID;
```

```
    }

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

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

    //setter or mutator method for client name and balance amount
    public void setClientName(String clientName)
    {
        this.clientName = clientName;
    }

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

```
    //display method
    public void display()
    {
        System.out.println("Your Card: " + cardID);
        // check if the client name is empty or not
        if(clientName == ""){
            System.out.println("The client name is empty."); //message when client name is unassigned
        }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);
    }
}
```

- **DebitCard**

```
/**
 * Write a description of class Debitcard here.
 *
 * @author (22015816 Samir Gurung)
 * @version (1.0.0)
 */
public class DebitCard extends BankCard
{
    //attributes for DebitCard class
    private int pinNumber;
    private int withdrawalAmount;
    private String dateOfWithdrawal;
    private boolean hasWithdrawn;

    //constructor
    public DebitCard(double balanceAmount, int cardID, String bankAccount, String issuerBank, String clientName,int pinNumber)
    {
        super(balanceAmount, cardID, bankAccount, issuerBank);
```

```
        super.setClientName(clientName);
        this.pinNumber = pinNumber;
        this.hasWithdrawn = false;

    }



    //getter or accessor method for all the attributes
    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 or mutator method for withdrawal amount
    public void setWithdrawalAmount(int withdrawalAmount)
    {
        this.withdrawalAmount = withdrawalAmount;
    }

    //withdraw method
    public void withdraw(int withdrawalAmount, String dateOfWithdrawal, int pinNumber)
    {
        // check if the pin number is same or not
        if(this.pinNumber == pinNumber){
            //checks if withdrawal amount is less than or equal to balance amount
            if(withdrawalAmount<=super.getBalanceAmount()){
                //deducts withdrawal amount from balance amount and store in a variable
                double newBalanceAmount = super.getBalanceAmount() - withdrawalAmount;
                //set new balance amount in super class
                super.setBalanceAmount(newBalanceAmount);
                this.hasWithdrawn = true;
```

```
            this.withdrawalAmount = withdrawalAmount;
            this.dateOfWithdrawal = dateOfWithdrawal;
            System.out.println("Money successfully withdrawn! Your remaining balance is: " + super.getBalanceAmount());
        }else{
            System.out.println("Insufficient balance. Please try again.");
        }
    }else{
        System.out.println("PIN number is invalid.");
    }
}
//display method
public void display()
{
    //display the super class
    super.display();
    //checks if the transaction was successful or not and prints as required
    if(this.hasWithdrawn == true){
        System.out.println("Pin Number: " + pinNumber);
        System.out.println("Withdrawal Amount: " + withdrawalAmount);
        System.out.println("Date of Withdrawal: " + dateOfWithdrawal);
    }else{
```

```
        System.out.println("Transaction was unsuccessful. Your balance is: "+ super.getBalanceAmount());
    }
  }
}
```

- **CreditCard**

```
/**
 * Write a description of class CreditCard here.
 *
 * @author (22015816 Samir Gurung)
 * @version (a version number or a date)
 */
public class CreditCard extends BankCard
{
    //attributes
    private int cvcNumber;
    private double creditLimit;
    private double interestRate;
    private String expirationDate;
    private int gracePeriod;
    private boolean isGranted;

    //constructor
    public CreditCard(int cardID,String clientName, String issuerBank, String bankAccount,
    double balanceAmount, int cvcNumber, double interestRate, String expirationDate)
```

```
    {
        //call made to superclass constructor with 4 parameters and a setter method
        super(balanceAmount,cardID, bankAccount,issuerBank);
        this.setClientName(clientName);
        //assigns parameter values and a false value
        this.cvcNumber = cvcNumber;
        this.interestRate = interestRate;
        this.expirationDate = expirationDate;
        this.isGranted = false;
    }

    //getter or accessor method for all attributes
    public int getCvcNumber()
    {
        return this.cvcNumber;
    }

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

    public double getInterestRate()
```

```
    {
        return this.interestRate;
    }

    public String getExpirationDate()
    {
        return this.expirationDate;
    }

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

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

    //method to set the credit limit
    public void setCreditLimit(double newCreditLimit, int newGracePeriod)
    {
        this.creditLimit = newCreditLimit;
```

```
        this.gracePeriod = newGracePeriod;
        //credit limit should be less or equals to 2.5 times the balance amount of super class
        if(creditLimit <= 2.5 * (super.getBalanceAmount())){
            System.out.println("Credit is successfully granted!");
            this.isGranted = true;
        }else{
            System.out.println("Credit limit exceeded, hence credit could not be granted.");
        }
    }

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

    //display method
    public void display()
    {
```

```
        //display the super class display()
        super.display();
        if(isGranted == true){
            System.out.println("Credit limit is valid, hence credit is granted!");
            System.out.println("Credit Limit:" + this.creditLimit);
            System.out.println("Grace Period:" + this.gracePeriod);
        }else{
            System.out.println("Credit is not granted.");
        }
        System.out.println("CVC number:" + this.cvcNumber);
        System.out.println("Interest Rate:" + this.interestRate);
        System.out.println("Expiration Date:" + this.expirationDate);
    }
}
```

- **BankGUI**

```
/**
 * Write a description of class BankGUI here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
import java.util.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.color.*;
public class BankGUI implements ActionListener
{
   private JLabel heading, cardIdLabel, clientNameLabel, issuerBankLabel, bankAccountLabel,balanceAmountLabel, cardIdLabel2;
   private JTextField cardIdTextField, clientNameTextField, issuerBankTextField, bankAccountTextField, balanceAmountTextField, cardIdTextField2;
   private JButton displayBtn, clearBtn;
   // Debit Card components
```

```
// Label, textfield, combobox and buttons needed for debitcard

private JLabel pinNumberLabel, withdrawalAmountLabel,withdrawalDateLabel, pinNumberLabel2;
private JTextField pinNumberTextField, withdrawalAmountTextField, pinNumberTextField2;

//For withdrawal date

private JComboBox dayOfWithdrawalComboBox, monthOfWithdrawalComboBox, yearOfWithdrawalComboBox;
private JButton addDebitCardButton, withdrawBtn;

// For Credit Card components
// Label
private JLabel cvcNumberLabel, creditLimitLabel, interestRateLabel, gracePeriodLabel, expirationDateLabel,CcardIdLabel,CissuerBankLabel,CclientNameLabel,CbankAccountLabel,CbalanceAmountLabel,cardIdLabel4;
private JTextField cvcNumberTextField, creditLimitTextField, interestRateTextField, gracePeriodTextField,CcardIdTextField,CissuerBankTextField,CclientNameTextField,CbankAccountTextField,CbalanceAmountTextField,cardIdTextField4;
private JComboBox dayOfExpirationComboBox, monthOfExpirationComboBox, yearOfExpirationComboBox;
private JButton addCreditCardBtn, setCreditLimitBtn, cancelCreditCardBtn;
```

```
    private JFrame frame;
    ArrayList<BankCard> cards = new ArrayList<BankCard>();

    public BankGUI(){
        //Frame code
        JFrame frame = new JFrame("Bank OF KATHMANDU");
        heading = new JLabel("BANK OF KATHMANDU");
        heading.setBounds(320,2,200,100);
        frame.add(heading);

        //Card id
        cardIdLabel = new JLabel("Card ID :");
        cardIdLabel.setBounds(80,120,50,20);
        cardIdTextField = new JTextField();
        cardIdTextField.setBounds(142,120,60,25);

        cardIdLabel2 = new JLabel("Card ID :");
        cardIdLabel2.setBounds(80, 570, 60, 19);
        cardIdTextField2 = new JTextField();
        cardIdTextField2.setBounds(142, 570, 60, 25);

        //Issuer bank
        issuerBankLabel = new JLabel("Issuer Bank :");
```

```
issuerBankLabel.setBounds(53,170,84,14);
issuerBankTextField = new JTextField();
issuerBankTextField.setBounds(142,170,125,25);

//Bank Account

bankAccountLabel = new JLabel("Bank Account :");
bankAccountLabel.setBounds(45,264,101,31);
bankAccountTextField = new JTextField();
bankAccountTextField.setBounds(140,220,125,25);

//Client Name

clientNameLabel = new JLabel("Client Name :");
clientNameLabel.setBounds(52,202,101,55);
clientNameTextField = new JTextField();
clientNameTextField.setBounds(142,269,125,25);

//Balance Amount

balanceAmountLabel = new JLabel("Balance Amount :");
balanceAmountLabel.setBounds(30,321,113,25);
balanceAmountTextField = new JTextField();
```

```
balanceAmountTextField.setBounds(142,321,125,25);

//Pin

pinNumberLabel = new JLabel("PIN :");
pinNumberLabel.setBounds(100,370,34,14);
pinNumberTextField = new JTextField();
pinNumberTextField.setBounds(142,368,52,25);
pinNumberLabel2 = new JLabel("PIN :");
pinNumberLabel2.setBounds(105, 690, 60, 19);
pinNumberTextField2 = new JTextField();
pinNumberTextField2.setBounds(142, 690, 60, 25);

//Withdraw Amount

withdrawalAmountLabel = new JLabel("Withdraw Amount :");
withdrawalAmountLabel.setBounds(28,610,124,22);
withdrawalAmountTextField = new JTextField();
withdrawalAmountTextField.setBounds(142,610,125,25);

//Withdraw Date
withdrawalDateLabel = new JLabel("Withdrawal Date :");
withdrawalDateLabel.setBounds(35, 650, 112, 18);
```

```
String[] dayOfWithdrawal = {"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"};
dayOfWithdrawalComboBox = new JComboBox(dayOfWithdrawal);
dayOfWithdrawalComboBox.setBounds(255,650,55,25);

String[] monthOfWithdrawal = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};
monthOfWithdrawalComboBox = new JComboBox (monthOfWithdrawal);
monthOfWithdrawalComboBox.setBounds(200,650,55,25);

String[] yearOfWithdrawal = {"1990","1991","1992","1993","1994","1995","1996","1997","1998","1999","2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010","2011","2012","2013","2014","2015","2016"};
yearOfWithdrawalComboBox = new JComboBox(yearOfWithdrawal);
yearOfWithdrawalComboBox.setBounds(142,650,55,25);

//For Debit Card Buttons
addDebitCardButton = new JButton("Add Debit Card");
addDebitCardButton.setBounds(25, 510, 126, 22);
withdrawBtn = new JButton("WithDraw");
withdrawBtn.setBounds(210,690,100,25);
addDebitCardButton.addActionListener(this);
```

```
withdrawBtn.addActionListener(this);

// Now for Credit card

// card id
CcardIdLabel = new JLabel("Card ID : ");
CcardIdLabel.setBounds(484,120,65,21);
CcardIdTextField = new JTextField();
CcardIdTextField.setBounds(555,120,60,25);
// issuerbank
CissuerBankLabel = new JLabel("Issuer Bank :");
CissuerBankLabel.setBounds(460,170,84,14);
CissuerBankTextField = new JTextField();
CissuerBankTextField.setBounds(556,168,126,25);

//Client Name
CclientNameLabel = new JLabel("Client Name :");
CclientNameLabel.setBounds(457,210,110,31);
CclientNameTextField = new JTextField();
CclientNameTextField.setBounds(556,214,127,25);

// bank account
CbankAccountLabel = new JLabel("Bank Account :");
```

```
CbankAccountLabel.setBounds(451,249,101,55);
CbankAccountTextField = new JTextField();
CbankAccountTextField.setBounds(556,267,127,25);

//balance amount
CbalanceAmountLabel = new JLabel("Balance Amount :");
CbalanceAmountLabel.setBounds(435,312,113,20);
CbalanceAmountTextField = new JTextField();
CbalanceAmountTextField.setBounds(556,312,129,25);

// CVC number
cvcNumberLabel = new JLabel("CVC : ");
cvcNumberLabel.setBounds(502,330,43,75);
cvcNumberTextField = new JTextField();
cvcNumberTextField.setBounds(556,355,55,25);

// For Interest Rate

interestRateLabel = new JLabel("Interest Rate :");
interestRateLabel.setBounds(454,400,88,21);
interestRateTextField = new JTextField();
interestRateTextField.setBounds(557,400,55,25);
```

```
// For Expiration Date

expirationDateLabel = new JLabel("Expiration Date :");
expirationDateLabel.setBounds(438,450,106,21);
String[] dayOfExpiration = {"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"};
dayOfExpirationComboBox = new JComboBox(dayOfExpiration);
dayOfExpirationComboBox.setBounds(686,450,55,25);

String[] monthOfExpiration = {"January","February","March","April","May","June","July","August","September","october","November","December"};
monthOfExpirationComboBox = new JComboBox(monthOfExpiration);
monthOfExpirationComboBox.setBounds(620,450,55,25);

String[] yearOfExpiration = {"1990","1991","1992","1993","1994","1995","1996","1997","1998","1999","2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010","2011","2012","2013","2014","2015","2016","2017"};
yearOfExpirationComboBox = new JComboBox(yearOfExpiration);
yearOfExpirationComboBox.setBounds(556,450,55,25);

// CardId for Credit
```

```
cardIdLabel4 = new JLabel("Card ID :");
cardIdLabel4.setBounds(484,570,60,19);
cardIdTextField4 = new JTextField();
cardIdTextField4.setBounds(556,570,60,25);

//For Credit Limit
creditLimitLabel = new JLabel("Credit Limit :");
creditLimitLabel.setBounds(464,585,99,73);
creditLimitTextField = new JTextField();
creditLimitTextField.setBounds(556,610,60,25);

//For Grace Period
gracePeriodLabel = new JLabel("Grace Period :");
gracePeriodLabel.setBounds(455,650,99,25);
gracePeriodTextField = new JTextField();
gracePeriodTextField.setBounds(555,650,60,25);

addCreditCardBtn = new JButton("Add Credit Card");
addCreditCardBtn.setBounds(640,510,126,22);
cancelCreditCardBtn = new JButton ("Cancel");
cancelCreditCardBtn.setBounds(705,650,74,24);
setCreditLimitBtn = new JButton ("Set");
setCreditLimitBtn.setBounds(625,650,74,24);
```

```
addCreditCardBtn.addActionListener(this);
cancelCreditCardBtn.addActionListener(this);
setCreditLimitBtn.addActionListener(this);

/* Credit Card's GUI code is finished here */

/* Button to display and clear the items in the array*/
clearBtn = new JButton("Clear");
clearBtn.setBounds(560,725,124,29);
displayBtn = new JButton("Display");
displayBtn.setBounds(420,725,124,29);
clearBtn.addActionListener(this);

//Frame code
frame.setSize(800,800);
frame.setResizable(false);
frame.setLayout(null);
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.add(cardIdLabel);
frame.add(cardIdTextField);
frame.add(issuerBankLabel);
frame.add(issuerBankTextField);
```

```
frame.add(bankAccountLabel);
frame.add(bankAccountTextField);
frame.add(clientNameLabel);
frame.add(clientNameTextField);
frame.add(balanceAmountLabel);
frame.add(balanceAmountTextField);
frame.add(pinNumberLabel);
frame.add(pinNumberTextField);
frame.add(withdrawalAmountLabel);
frame.add(withdrawalAmountTextField);
frame.add(withdrawalDateLabel);
frame.add(dayOfWithdrawalComboBox);
frame.add(monthOfWithdrawalComboBox);
frame.add(yearOfWithdrawalComboBox);
frame.add(addDebitCardButton);
frame.add(withdrawBtn);
frame.add(cvcNumberLabel);
frame.add(cvcNumberTextField);
frame.add(interestRateLabel);
frame.add(interestRateTextField);
frame.add(expirationDateLabel);
frame.add(dayOfExpirationComboBox);
frame.add(monthOfExpirationComboBox);
```

```
frame.add(yearOfExpirationComboBox);
frame.add(creditLimitLabel);
frame.add(creditLimitTextField);
frame.add(gracePeriodLabel);
frame.add(gracePeriodTextField);
frame.add(addCreditCardBtn);
frame.add(cancelCreditCardBtn);
frame.add(setCreditLimitBtn);
frame.add(clearBtn);
frame.add(displayBtn);
frame.add(cardIdLabel2);
frame.add(cardIdTextField2);
frame.add(pinNumberLabel2);
frame.add(pinNumberTextField2);
frame.add(cardIdLabel4);
frame.add(cardIdTextField4);

frame.add(CcardIdLabel);
frame.add(CcardIdTextField);
frame.add(CissuerBankLabel);
frame.add(CissuerBankTextField);
frame.add(CclientNameLabel);
frame.add(CclientNameTextField);
```

```
        frame.add(CbankAccountLabel);
        frame.add(CbankAccountTextField);
        frame.add(CbalanceAmountLabel);
        frame.add(CbalanceAmountTextField);

    }

    //Functionality for debit card
    public void actionPerformed(ActionEvent e){

        // Get all the values of the constructor parameters
        //if(e.getSource() == addDebitCardButton){
        if(e.getSource() == addDebitCardButton){
            try{
                // Retrieving values from text fields

                int cardId = Integer.parseInt(cardIdTextField.getText());
                String clientName = clientNameTextField.getText();
                String issuerBank = issuerBankTextField.getText();
                String bankAccount = bankAccountTextField.getText();
                double balanceAmount =
Double.parseDouble(balanceAmountTextField.getText());
                int pinNumber = Integer.parseInt(pinNumberTextField.getText());
```

```
            boolean toAdd = true;
            DebitCard dbCard;

            if(cards.isEmpty()){
                dbCard = new DebitCard(balanceAmount, cardId, bankAccount, issuerBank, clientName, pinNumber);
                cards.add(dbCard);
                JOptionPane.showMessageDialog(frame, "Debit card is added successfully.", "Alert", JOptionPane.INFORMATION_MESSAGE);

            }else{
                for(BankCard card : cards){
                    if(card instanceof DebitCard){

                        dbCard = (DebitCard) card;
                        dbCard.display();

                        if(dbCard.getCardID() == cardId){

                            JOptionPane.showMessageDialog(frame, "Debit card already exists.", "Error", JOptionPane.ERROR_MESSAGE);
                            toAdd = false;
                            break;
```

```
                }
            }else{

                JOptionPane.showMessageDialog(frame, "Card Cannot Be Added.", "Error", JOptionPane.ERROR_MESSAGE);
                toAdd = false;
                break;

            }

            if(toAdd == true){

                dbCard = new DebitCard(balanceAmount, cardId, bankAccount, issuerBank, clientName, pinNumber);
                cards.add(dbCard);
                JOptionPane.showMessageDialog(frame, "Debit Card Is Added Successfully.","Alert.", JOptionPane.INFORMATION_MESSAGE);

            }
        }

    }
```

```
        }catch(NumberFormatException ex){
            JOptionPane.showMessageDialog(frame, "Invalid !! \n Please input correct data", "Input Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    // Functionality for adding Credit Card button

    if(e.getSource() == addCreditCardBtn){

        try{

            // retrieving values from text fields

            int cardId = Integer.parseInt(CcardIdTextField.getText());
            String clientName = CclientNameTextField.getText();
            String issuerBank = CissuerBankTextField.getText();
            String bankAccount = CbankAccountTextField.getText();
            double balanceAmount = Double.parseDouble(CbalanceAmountTextField.getText());
            int cvcNumber = Integer.parseInt(cvcNumberTextField.getText());
            double interestRate = Double.parseDouble(interestRateTextField.getText());
            String Day = (String) dayOfExpirationComboBox.getSelectedItem();
```

```
String Month = (String) monthOfExpirationComboBox.getSelectedItem();
String Year = (String) yearOfExpirationComboBox.getSelectedItem();
String expirationDate = Day + " " + Month + " " + Year;
boolean toAdd = true;
CreditCard ggCard;

if(cards.isEmpty()){
    ggCard = new CreditCard(cardId ,clientName, issuerBank, bankAccount, balanceAmount, cvcNumber, interestRate, expirationDate);
    cards.add(ggCard);
    JOptionPane.showMessageDialog(frame, "Credit Card Has Been Added Successfully. ", "Alert", JOptionPane.INFORMATION_MESSAGE);

}
else{

    for(BankCard card : cards){

        if(card instanceof CreditCard){

            ggCard = (CreditCard) card;
            if(ggCard.getCardID() == cardId){
```

```
                    JOptionPane.showMessageDialog(frame, "Credit Card Already Exists.", "Error", JOptionPane.ERROR_MESSAGE);
                    toAdd = false;
                    break;
                }
            }

            else{

                JOptionPane.showMessageDialog(frame, "Card Not Found.", "Error", JOptionPane.ERROR_MESSAGE);
                toAdd = false;
                break;
            }

        }
        if(toAdd == true){

            ggCard = new CreditCard(cardId ,clientName, issuerBank, bankAccount, balanceAmount, cvcNumber, interestRate, expirationDate);
            cards.add(ggCard);
            JOptionPane.showMessageDialog(frame, "Credit Card Is Added Successfully.", "Alert", JOptionPane.INFORMATION_MESSAGE);
        }
```

```
            }
        }
        catch(NumberFormatException ex){

            JOptionPane.showMessageDialog(frame, "Invalid !! \n Please Input Correct Data.", "Input Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    // Setting CreditLimit

    if(e.getSource() == setCreditLimitBtn){
        try{
            //Retrieving values from text fields

            int cardId = Integer.parseInt(cardIdTextField4.getText());
            double newCreditLimit = Double.parseDouble(creditLimitTextField.getText());
            int newGracePeriod = Integer.parseInt(gracePeriodTextField.getText());

            if(cards.isEmpty()){
                JOptionPane.showMessageDialog(frame, "Card Has Not Been Added", "\n Please Add The Card First", JOptionPane.ERROR_MESSAGE);
```

```
        }

        else{

            //If credit card is already present

            for(BankCard card:cards){
                //Check if its credit card or not

                if(card instanceof CreditCard){

                    //Downcasting

                    CreditCard ggCard = (CreditCard) card;

                    if(ggCard.getCardID() == cardId){

                        //Calling set credit limit method

                        ggCard.setCreditLimit(newCreditLimit, newGracePeriod);

                        if(newCreditLimit <= 2.5 * ggCard.getBalanceAmount()){
```

```
                    JOptionPane.showMessageDialog(frame, "Credit Limit Has Been
Set : \n Credit Limit :" + newCreditLimit + "\n Grace Period :" + newGracePeriod, "Alert
!!", JOptionPane.INFORMATION_MESSAGE);


                }
                else{


                    JOptionPane.showMessageDialog(frame, "Credit Limit Has Not
Been Set", "Error", JOptionPane.ERROR_MESSAGE);


                }
            }
            else{
                JOptionPane.showMessageDialog(frame, "Your CardID Is Incorrect
!!", "Error", JOptionPane.ERROR_MESSAGE);


            }


        }
        else{


            JOptionPane.showMessageDialog(frame, "Card Not Found !!", "Error",
JOptionPane.ERROR_MESSAGE);
        }
```

```
            }
        }

    }
    catch(NumberFormatException ex){

        JOptionPane.showMessageDialog(frame, "Invalid Input !!, Please Check Again", "Error !!", JOptionPane.INFORMATION_MESSAGE);
    }

}
// Setting Cancle button
if(e.getSource() == cancelCreditCardBtn){
    try{
        //Retreving calues from textFields

        int cardId = Integer.parseInt(cardIdTextField4.getText());
        int newcreditLimit = Integer.parseInt(creditLimitTextField.getText());
        int newgracePeriod = Integer.parseInt(gracePeriodTextField.getText());
        int cvcNumber = Integer.parseInt(cvcNumberTextField.getText());

        if(cards.isEmpty()){
```

```
                JOptionPane.showMessageDialog(frame, "Card Not Found!!", "Error", JOptionPane.ERROR_MESSAGE);

            }
            else{

                for (BankCard card : cards){

                    if(card instanceof CreditCard){

                        //Downcasting

                        CreditCard ggCard = (CreditCard) card;

                        if(ggCard.getCardID() == cardId){

                            ggCard.cancelCreditCard();
                            creditLimitTextField.setText("");
                            gracePeriodTextField.setText("");
                            cvcNumberTextField.setText("");

                            JOptionPane.showMessageDialog(frame, "Credit Card Is Cancelled Successfully", "No Error Found", JOptionPane.INFORMATION_MESSAGE);
```

```
                    }
                    else{
                        JOptionPane.showMessageDialog(frame,"Incorrect CardID. Credit Card Is Not Cancelled","Error", JOptionPane.ERROR_MESSAGE);
                    }
                }

                else{
                    JOptionPane.showMessageDialog(frame,"Credit Card Not Found", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }

        }

    }
    catch(NumberFormatException ex){
        JOptionPane.showMessageDialog(frame,"Invalid Input!!Please Check Again", "Error",JOptionPane.INFORMATION_MESSAGE);

    }
}
```

```
// Implementing Action Listener to the WithdrawButton

if(e.getSource() == withdrawBtn){
    try{

        //Retrieving values form text field

        int cardId = Integer.parseInt(cardIdTextField2.getText());
        int withdrawalAmount = Integer.parseInt(withdrawalAmountTextField.getText());
        int pinNumber = Integer.parseInt(pinNumberTextField2.getText());
        String dbDay = (String) dayOfWithdrawalComboBox.getSelectedItem();
        String dbMonth = (String) monthOfWithdrawalComboBox.getSelectedItem();
        String dbYear = (String) yearOfWithdrawalComboBox.getSelectedItem();
        String dateOfWithdrawal = dbDay + " " + dbMonth + " " + dbYear;

        if(cards.isEmpty()){

            JOptionPane.showMessageDialog(frame, "Card Has Not Been Added" , "\nPlease Add The Card First", JOptionPane.ERROR_MESSAGE);

        }
        //If Debit card is already present
```

```
        else{
            //loop through the arraylist

            for(BankCard card : cards){
                //Its Debit Card or not?
                if(card instanceof DebitCard){
                    //Downcasting

                    DebitCard dbCard = (DebitCard) card;

                    if(dbCard.getCardID() == cardId && dbCard.getPinNumber() ==
pinNumber){
                        double initialAmount = dbCard.getBalanceAmount();
                        dbCard.withdraw(withdrawalAmount, dateOfWithdrawal,
pinNumber);
                        double finalAmount = dbCard.getBalanceAmount();
                        double remainingAmount = initialAmount - withdrawalAmount;

                        if(withdrawalAmount <= initialAmount){

                            JOptionPane.showMessageDialog(frame, "Money Is Successfully
Withdrawn. Your Remaining Balance Is : ","Withdraw success",
JOptionPane.INFORMATION_MESSAGE);
```

```
                    }

                    else{

                        JOptionPane.showMessageDialog(frame, "Your Balance Is Insufficient", "Error", JOptionPane.ERROR_MESSAGE);
                    }

                }
                else{
                    JOptionPane.showMessageDialog(frame,"Your PIN Or CardID IS Incorrect", "Error", JOptionPane.ERROR_MESSAGE);
                }

            }
            else{

                JOptionPane.showMessageDialog(frame," Debit Card Not Found", "Error",JOptionPane.ERROR_MESSAGE);

            }
```

```
                }
            }
        }catch(NumberFormatException ex){
            JOptionPane.showMessageDialog(frame," Invalid Input", "Please Try Again",JOptionPane.INFORMATION_MESSAGE );


        }
    }


    if(e.getSource() == clearBtn){
        cardIdTextField.setText("");
        clientNameTextField.setText("");
        issuerBankTextField.setText("");
        bankAccountTextField.setText("");
        balanceAmountTextField.setText("");
        cardIdTextField2.setText("");
        pinNumberTextField.setText("");
        withdrawalAmountTextField.setText("");
        pinNumberTextField2.setText("");
        cvcNumberTextField.setText("");
        creditLimitTextField.setText("");
        interestRateTextField.setText("");
        gracePeriodTextField.setText("");
```

```
            CcardIdTextField.setText("");
            CissuerBankTextField.setText("");
            CclientNameTextField.setText("");
            CbankAccountTextField.setText("");
            CbalanceAmountTextField.setText("");
            cardIdTextField4.setText("");

        }

    //Display button
        if(e.getSource() == displayBtn){
            int cardId = Integer.parseInt(cardIdTextField.getText());
            String clientName = clientNameTextField.getText();
            String issuerBank = issuerBankTextField.getText();
            String bankAccount = bankAccountTextField.getText();
            double balanceAmount =
Double.parseDouble(balanceAmountTextField.getText());
            int pinNumber = Integer.parseInt(pinNumberTextField.getText());

            int withdrawalAmount = Integer.parseInt(withdrawalAmountTextField.getText());
            String day = (String) dayOfWithdrawalComboBox.getSelectedItem();
            String month = (String) monthOfWithdrawalComboBox.getSelectedItem();
            String year = (String) yearOfWithdrawalComboBox.getSelectedItem();
```

```
String withdrawalDate = day+""+month+""+year;


int CcardId = Integer.parseInt(CcardIdTextField.getText());
String CclientName = CclientNameTextField.getText();
String CissuerBank = CissuerBankTextField.getText();
String CbankAccount = CbankAccountTextField.getText();
String CbalanceAmount = CbalanceAmountTextField.getText();
int cvcNumber = Integer.parseInt(cvcNumberTextField.getText());
double interestRate = Double.parseDouble(interestRateTextField.getText());
String day1 = (String) dayOfExpirationComboBox.getSelectedItem();
String month1 = (String) monthOfExpirationComboBox.getSelectedItem();
String year1 = (String) yearOfExpirationComboBox.getSelectedItem();
String expirationDate = day1+""+month1+""+year1;


double nreCreditLimit = Double.parseDouble(creditLimitTextField.getText());
int newGracePeriod = Integer.parseInt(gracePeriodTextField.getText());


if(cards.isEmpty()){
    JOptionPane.showMessageDialog(frame, "No card Found.","Error",JOptionPane.ERROR_MESSAGE);


}
else{
```

```
            for(BankCard  debObj1 : cards){
                if(debObj1 instanceof DebitCard){
                    DebitCard debitCard = (DebitCard) debObj1;
                    debObj1.display();

                    String dis = " Card ID : "+cardId+"\nClient Name:"+clientName+"\nIssuer
Bank:"+issuerBank+"\nBank                    Account"+bankAccount+"\nBalance
Amount:"+balanceAmount+"\nPin Number:"+pinNumber;
                    JOptionPane.showMessageDialog(frame,  dis,"Debit  card  Displayed",
JOptionPane.INFORMATION_MESSAGE);




                }
            }
        }
    }

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

```
// write the logic of the button functionality here
// add debit card
// get all the values
// check all the values
// if values are not correct: show message

// if array list is empty:
// Add Debit card
// call a constructor
// create object of debit card
// add object to the arraylist
// show message:
//else
// if debit card is already present:
// loop through the arraylist,if id is same
// dont add
```

```
// show message
// else:
// Add debit card
// call a constructor
// create object of debit card
// add object to the arraylist
// show message
// WITHDRAW button
// get the parameters
// call the withdraw method

// if the arraylist is empty:
// dont withdraw
// show message:

// else
// check id the debitcard is present
// loop thorough the arraylist
// if it si debitcard or not?
// id card id matches
//  show information in a dialog box
// if PIN number matches
// call the withdraw method with the parameters
```

```
// show message: Amount has been withdrawn
// else
// show message : PIN does not match
// else
// show message: card not found


// Display method (everything)
// call the display method


// if arraylist is empty:
// dont display
// show message: empty  | nothing to display
// else:
// loop through the arraylist
// if debit card is present (if object instance of Debit card)
//downcast it
// call the display method
//else
// show message: debit card not found


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

//Step1: Create JFrame using constructor

//Use the method of java frame:

//step2: Set the size of the JFrame