



**informatics**  
college • pokhara



**LONDON  
METROPOLITAN  
UNIVERSITY**

**Module Code & Module Title**

**Programming**

**CS4001NP**

**Assessment weightage & Type**

**30% Individual Coursework - 3**

**Year and semester**

**2022-2023 Autumn**

**Name: Abishkar Chapagain**

**Group : C4**

**London met ID: 22068942**

**College ID:**

**Assignment due date: April 3, 2023**

**Assignment submission date: May 10, 2023**

**Declaration**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submission will be treated as non-submission and a mark of zero will be awarded.*

## Table of Contents

1. Introduction:	2
1. Class Diagram:	3
2.1 Class diagram of BankCard	4
2.2 Class diagram of Debit Card	5
2.3 Class diagram of Credit Card:	6
2.4 Class Diagram of Bank GUI	6
3. Pseudo Code	8
3.1 Pseudocode for Bank Card Class	8
3.2 Pseudocode for Debit Card Class	11
3.3 Pseudocode for Credit Card	14
3.4 Pseudo code for MyCustomButton	18
3.5 Pseudo Code for MyCustomTextField	19
3.6 Pseudo code for CustomComboBox	19
3.7 Pseudo code for MyCustomLabel	20
3.8 Pseudocode For ADD CREDIT CARD	21
3.9 Pseudocode For Add Debit Card	32
3.10 Pseudo Codes for CancelCreditCard	39
3.11 Pseudo Code for DisplayCreditCard	41
3.12 Pseudo Codes for DisplayDebitCard	43
3.13 Pseudocode For Bank GUI	49
3.14 Pseudocode for mainDisplay	51
3.15 Pseudo Code For SetCreditLimit	53
3.16 Pseudo Code for Withdraw From Debit Card	59
4. METHOD DESCRIPTION	66
4.1 Method description do CustomComboBox	66
4.2 Method Descripton For MyCustomLabel	68
4.3 Method Description For My Custom Text Field	68
4.4 Method Description For AddCreditCard	69
4.5 Method Description For AddDebitCard	70
4.6 Method Description For Cancel Credit Card	71

---

4.7 Method Description For Display Credit Card .....	72
4.8 Method Description For Display Debit Card .....	73
4.9 Method description for Set Credit Limit .....	74
4.10 Method description For Main .....	75
4.11 Method Description For WithDraw From Debit Card .....	76
4.12 Method Description for Bank Card .....	77
4.13 Method Description of Debit Card.....	77
4.14 Method Description for Credit Card .....	78
4.15 Method Description For Bank GUI .....	79
5. Testing.....	80
5.1 Test 1 : Using the command prompt, Testing if the program can be built and executed: .....	80
6 Test Table 1 .....	81
A. Main Page .....	82
B. Add to debit card .....	82
7 Test Table 2: .....	84
C. Add Credit Card.....	85
Display the details of Credit Card .....	86
8 Test Table 3 .....	88
D. Withdraw From Debit Card.....	88
Display After Withdraw.....	89
9 Test Table 4 .....	90
E. Set credit limit.....	90
F. Cancel Credit card .....	91
After cancelling drecit limit cvc number is being 0.....	92
10 Test Table 4 .....	94
6. Test 3.....	94
11 .....	95
6.1 Putting value error .....	95
6.2 Credit card.....	96
6.3 WITHDRAW .....	97
12 .....	97
6.4 Set Credit Limit.....	97
13 .....	98

---

13.5	cancel credit limit.....	98
14	.....	99
15	Error Detection and Correction .....	99
6.1	Syntax Error:.....	100
6.1.1	Error detection:.....	100
6.1.2	<b>Error correction:</b> .....	100
6.2	Runtime Error.....	101
6.2.1	Error Detection: .....	101
6.2.2	<b>Error Correction:</b> .....	102
6.3	Logical Error .....	103
6.3.1	Error Detection: .....	103
6.3.2	Error Correction .....	104
7.	Conclusion.....	105
8.	Appendix .....	106
8.1.	CustomComboBox.....	106
8.2	CustomButton .....	106
8.3	CustomLabel.....	107
8.4.	MyCustomTextField .....	107
8.5	Add Credit Card .....	107
8.6	Add Debit Card .....	112
8.7	Cancel Credit Card.....	115
8.8	Display Credit Card.....	117
8.9	Display Debit Card.....	118
8.10	main .....	119
8.11	Set Credit LI mit.....	120
8.12	With from debit card .....	123

---

Figure 1 Class Diagram OF Bank Card .....	4
Figure 2 Class Diagram of Debit Card.....	5
Figure 3 Class Diagram Of Credit Card .....	6
Figure 4 Class Diagram Of Bank GUI .....	7
Figure 5Class Diagram OF Whole Project.....	7
Figure 6Compiling and running program through command prompt.....	80
Figure 7 The GUI is displayed after compiling and running the program in the command prompt .....	81
Figure 8 displaying home .....	82
Figure 9 Adding Debit Card .....	83
Figure 10 Display After Adding Debit Card .....	84
Figure 11Adding credit Card .....	86
Figure 12 Display after adding Credit Card .....	87
Figure 13 Entering correct Pin number and Card Id to withdraw.....	88
Figure 14 Display after withdraw .....	89
Figure 15set credit limit .....	91
Figure 16canceling the credt limit .....	92
Figure 17 Details after canceling credit limit .....	93
Figure 18keeping int value less than 0 and mwssage was shown .....	95
Figure 19value error on credit card and message shown.....	96
Figure 20 Trying to withdraw more than monry then entered in debit card .....	97
Figure 21Entering negative in card id and message shown.....	98
Figure 22 Entering wrong Card Id in Cancel credit limit .....	99
Figure 23 Syntax Error occurred due to missing of semi-colon .....	100
Figure 24 Syntax error correction by adding semi-colon.....	101
Figure 25 something is missing in set layout .....	102
Figure 26 pRoblems on RunTime Error .....	102
Figure 27 Null was mising in SetLayout .....	103
Figure 28errann (>) sign .....	103
Figure 29 due to > sign withdrawn amount is >0 .....	104
Figure 30Logical error was missing and which was crrected .....	105

## 1. Introduction:

At the conclusion of our first year, on December 26, 2022, we received the second piece of coursework for our programming module. In this individual project, we will utilize an ArrayList and a new class to develop a graphical user interface (GUI) for a Banking management system.

For the project's initial stage, we created classes for bank cards, debit cards, and credit cards. We must now create a brand-new class called BankGUI, which will be able to hold an arraylist of Bank Card objects, including DebitCard and CreditCard. Despite having finished all of the prerequisites for the first part of the coursework, we still need to apply them using the context from the second coursework, which entails building a GUI.

As part of the coursework, we are required to provide text boxes where we may type in information such CardID, ClientName, Pin number, Issuer Bank, Withdrawal Amount, and other variables. A combo box where we may enter the date as well as many buttons that carry out particular tasks must also be included.

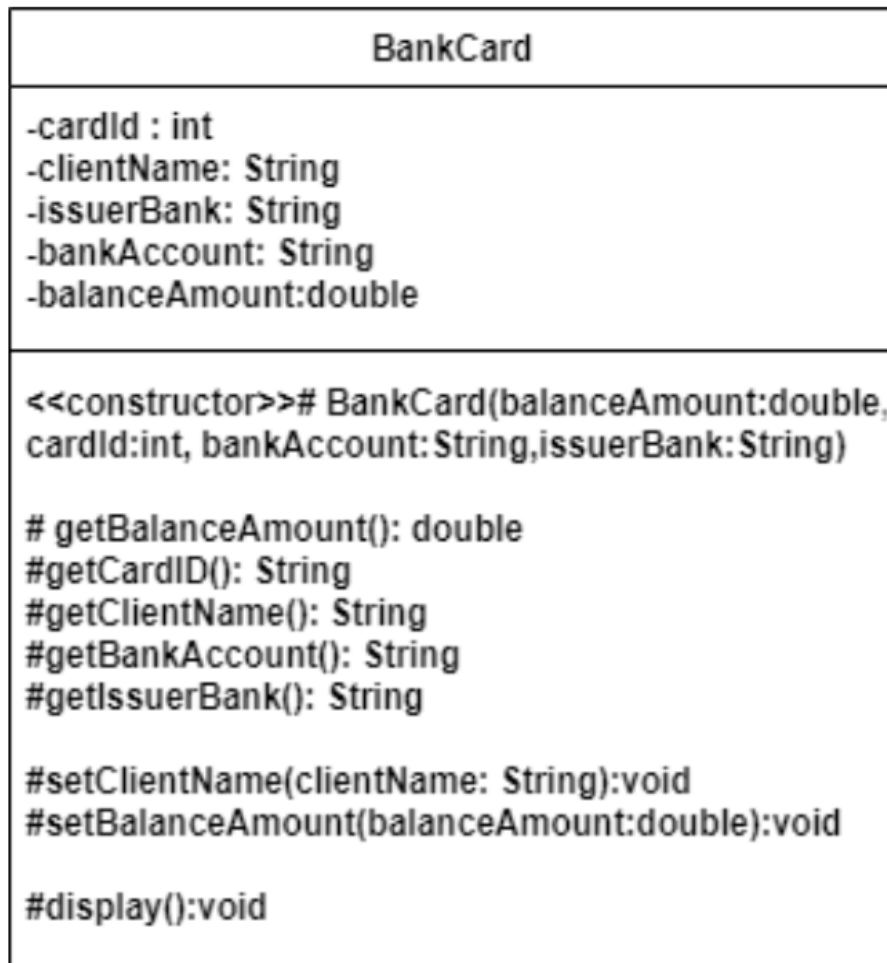
I used a variety of programs to do this coursework, including Draw.io for the class diagram, Ms. Word for the report writing, and IntelliJ IDEA for the coding. An integrated development environment called IntelliJ IDEA makes it easier to create applications using the Java programming language. A report is produced using Ms-Word, and the class diagram is made using the internet diagramming tool Draaw.io.

## 1. Class Diagram:

A class diagram is a schematic that outlines the classes, properties, actions, and connections between them inside a software system. Class name, characteristics, and other information are included in the class diagram in different parts.

methods. Codes are built for the construction of software applications with the use of class diagrams. Class diagrams are used to define the many types of system objects and the relationships that exist between them. This aids in providing a high level picture of an application. Even highly complicated information systems can have data models that are shown using class diagrams. Class diagrams are prepared prior to the start of programming in order to visualize the classes in the system .

## 2.1 Class diagram of BankCard

*Figure 1 Class Diagram OF Bank Card*



## 2.2 Class diagram of Debit Card

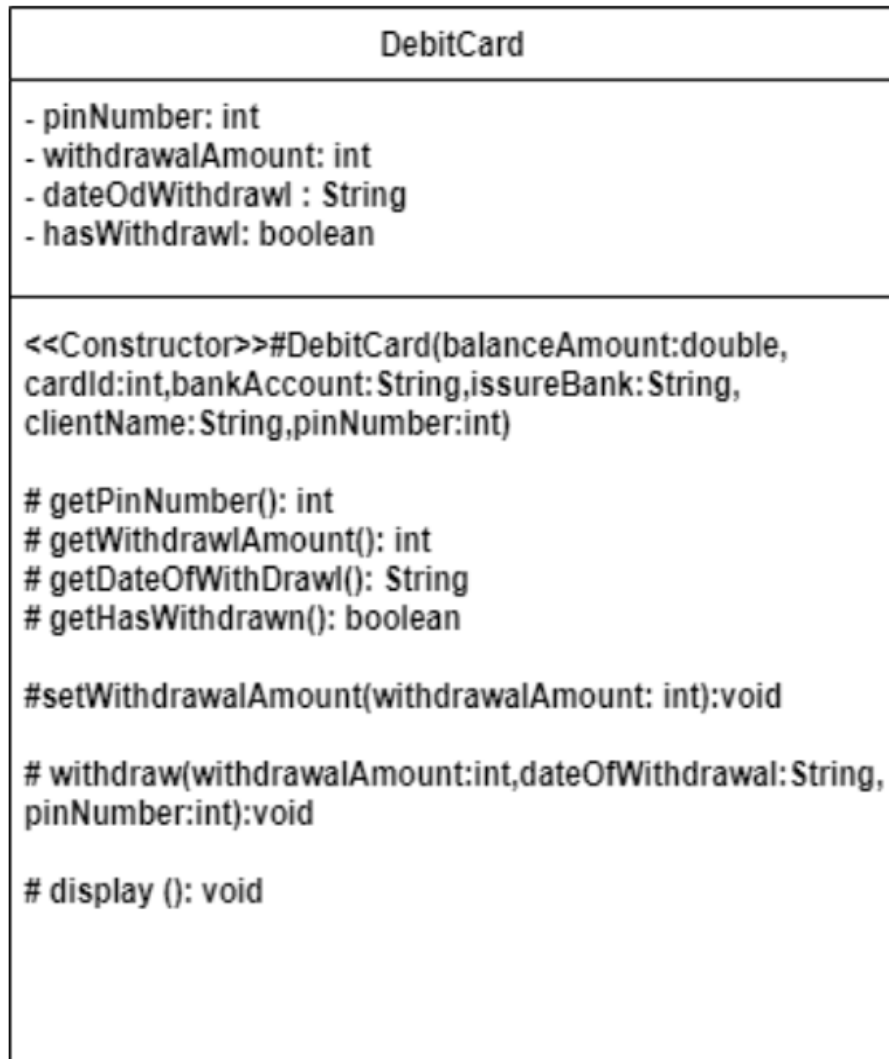


Figure 2 Class Diagram of Debit Card

### 2.3 Class diagram of Credit Card:



Figure 3 Class Diagram Of Credit Card

### 2.4 Class Diagram of Bank GUI

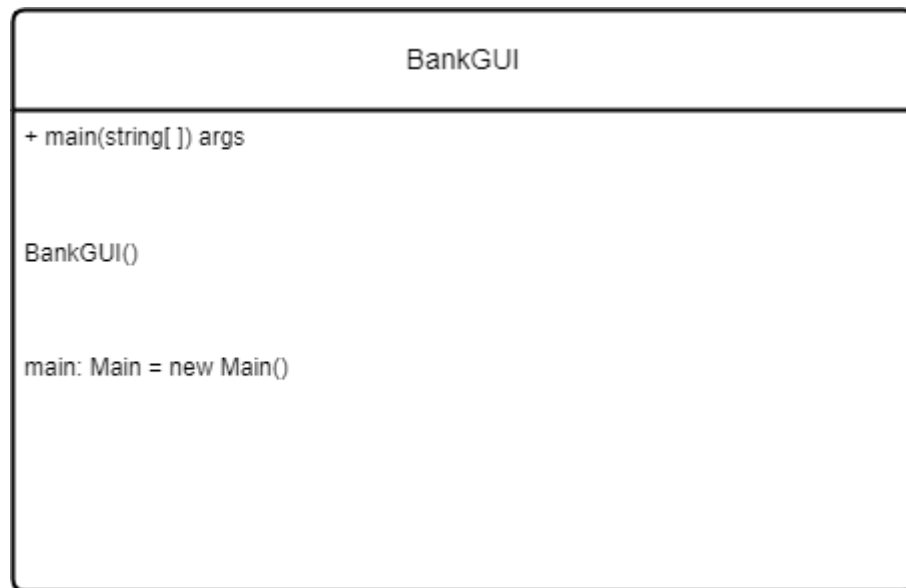


Figure 4 Class Diagram Of Bank GUI

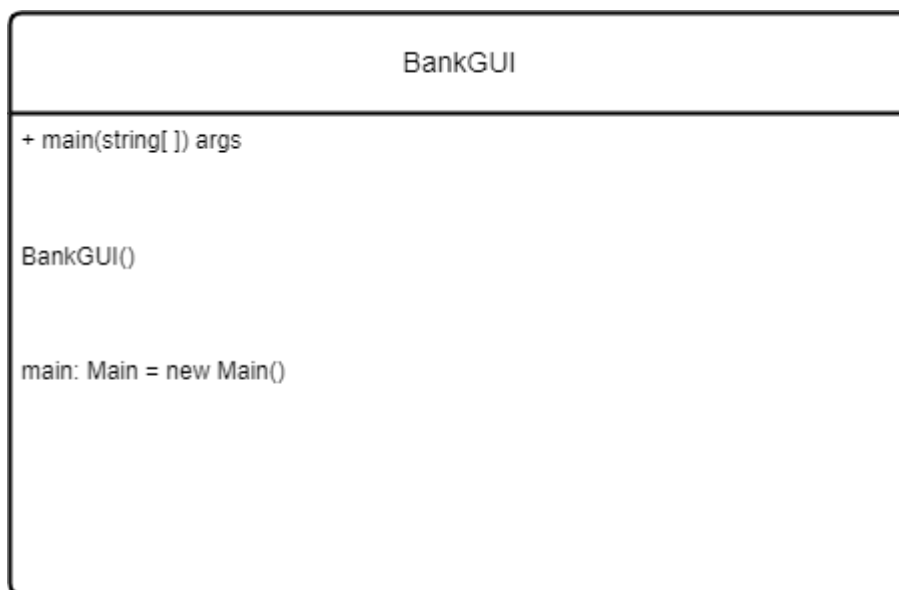


Figure 5 Class Diagram OF Whole Project

### 3. Pseudo Code

In disciplines that rely on algorithms and programming, the phrase "pseudo code" is frequently used. It is a mechanism that lets a programmer express how an algorithm is put into practice. Simply said, it is the fabricated representation of an algorithm. Because pseudo codes may be understood by programmers regardless of their programming experience or understanding, algorithms are frequently expressed using them. Pseudo code, as the name implies, is a fictitious version of code that even a layperson with little programming expertise can understand.

#### 3.1 Pseudocode for Bank Card Class

START

Create a Bank Card

Declare int cardID

Declare double balance Amount

Declare String bankAccount

Declare String clientName

Declare String issuerBank

Create Constructor for Bank Card(balanceAmount, card Id, bankAccount, issureBank)

Assign the value for balanceAmount

Assign the value for cardId

Assign the value for bankAccount

Assign the value for issueBank

Assign the empty value for clientName

Constructor has been ends here

Now accessor method starts

Create a getter method for CardID

Return cardId

Create a getter method for BalanceAmount

Return baanceAmount

Create a getter method for BankAccount

Return bankAmount

Create a getter method for CLientName

Return clientName

Create a getter method for IssureBank

Return issurebank

Accessor method ends here

Now setter method starts

Create a setter method for ClientName

Assign the value for clientName

Create setter method for balanceAmount

assign the value for balanceAmount

setter method ends here

Create display method

IF clientName==null || clientName isEmpty

THEN

Print "client name has not been set"

END IF

print cardId

print clientName

print issuerBank

print bankAccount

print balanceAmount

END

### 3.2 Pseudocode for Debit Card Class

Create class DebitCard extends BankCard

Declare int value pinNumber

Declare int value withdrawalAmount

Declare String value dateofWithdrawal

Declare Boolean value hasWithdrawn

Starts constructor here

Create a Constructor Debit Card (balance Amount ,cardId ,bankAccount ,issuerBank, clientName  
, pinNumber)

Call constructor of bank card with balanceAmount ,cardId ,bankAccount, issuerBank

Call Client Name from bankcard

Set pinNumber

Set hasWithdrawn to false

END constructor

Accessor method starts here

Create getter method getPinNumber

Return pinNumber.

Create getter method getWithdrawalAmount

Return WithdrawalAmount

Create getter method getDateOfWithdrawal

Return dateOfWithdrawn

Create getter method gethasWithdrawn

Return hasWithdrawn.

Accessor method ends here

Starts setter method here

Create a setter method for withdrawalAmount

Assign the value for withdrawal

Ends setter method here

Withdraw method starts

Create withdraw method (withdrawalAmount ,dateOfwithdrawal , pinNumber)

{Check pin number is valid or not using IF ELSE statement}

IF pinNumber is valid

{Again check withdrawalAmount is less than or equal to balanceAmount }

IF withdrawalAmount is less than or equal to balanceAmount

Set withdrawalAmount

Set dateOfWithdrawal

Set haswithdrawn to true

ELSE

Print" balance is insufficient"

ELSE

Print"Pin number is required"



Create display method

Call super display method

Print " pinNumber"

IF (baswithdrawn is true)

Print "withdrawalAmount"

Print "dateQfWithdrawal"

ELSE

Print "balance amount "

Print "no withdrawl has been made"

END

### 3.3 Pseudocode for Credit Card

START

Create class Credit Card extends Bank Card

Declare int cvcNumber

Declare double creditLimit

Declare double interestRate

Declare String expirationDate.

Declare int gracePeriod

Constructor starts here

Create Constructor for Credit Card (cardId, clientName, issuerBank ,bankAccount ,balanceAmount  
,cvcNumber ,interestRate, expiration Date)

Call constructor of Bank Card with (balanceAmount,cardId,bankAccount,issuerBank)

Set clientName using Setter method 'SetClientName'

Assign the value for cvcNumber

Assign the value for interestRate

Assign the value for expirationRate

Assign the value isGranted to false

End Constructor

Accessor method starts here

Create getter method getCyNumber.

return cvcNumber

Create getter method getCreditLimit

return creditLimit

Create getter method getinterestRate

return interestRate

Create getter method getExpirationRate

return expirationRate

Create getter method getGracePeriod

Return gracePeriod

Create getter method getIsGranted

return isGranted

accessor method ends here

setter method starts here

Create SetCreditLimit method(creditLimit, gracePeriod)

IF creditLimit is less than or equal to 2.5 times getBalanceAmount

Assign value for creditLimit

Assign value for gracePeriod

Assign value isGranted to true

ELSE

print"Credit cannot be issued"

Create display method

IF assign value of is granted is true

Call display method of bank card

Print cvcNumber.

Print set creditlimit

Print Credit cannot be issued"

Create display method

IF assign value of is granted is true

Call display method of bank card

Print cycNumber.

Print creditlimit

Print interestrate

Print expirationDate.

Print gracePeried to true

ELSE

Print "Credit cannot be issued"

Create method for cancel CreditCard

IF isGranted is false THEN

Initialize cvcNumber to zero

Initialize creditLimit to zero

Initialize graceperiod to zero

Initialize isgranted to false

ELSE

Print "credit has not been granted"

Create display method

IF is granted

THEN

Call the parent,s display using super

Print cvcNumber

Print Interest rate

Print gracePeriod

Print creditLimit

Print expirationDate

ELSE

Print "Credit has not been granted"

END

### 3.4 Pseudo code for MyCustomButton

Start

Define a class called MyCustomButton that extends JButton

Create a constructor for MyCustomButton that takes a string input parameter named "text"

Set the focusPainted property of the button to false to remove the focus border when the button is clicked

Set the contentAreaFilled property to true to enable the button to have a background color

Set the font of the button to "Tohama" and the size to 15

Set the foreground color of the button to white

Set the background color of the button to gray

Set the margin of the button to 10 pixels on all sides

End

### 3.5 Pseudo Code for MyCustomTextField

Start

Define a class called MyCustomTextField that extends JTextField

Create a constructor for MyCustomTextField that takes a string input parameter named "text"

Set the number of columns for the text field to 10

Set the border color of the text field to grey with a thickness of 2

Set the font of the text field to "Tohama" and the size to 15

End

### 3.6 Pseudo code for CustomComboBox

Start

Define a class called CustomComboBox that extends JComboBox

Create a constructor for CustomComboBox that takes a string input parameter named "values"

Set the font of the combo box to "Tohama" and the size to 15

Set the maximum row count of the combo box to 5

End

### 3.7 Pseudo code for MyCustomLabel

Start

Create a class named MyCustomLabel that extends JLabel

Create a constructor for the class MyCustomLabel that accepts a String input parameter named "text".

Set the font name Tohama and size 15

Set foreground to white

End



### 3.8 Pseudocode For ADD CREDIT CARD

Declare Package CourseWorkTwo Views

Import BankCard from Course WorkTwo Package

Import Custombutton,Customlabel,My CustomTestField

Import CreditCard form CourseorkTwoPackage

Import java.swing\*

Import java.awt\*

A Class AddCreditCard in created which extends JFrame

Initialize MyCustomLabel

cardIdLabel, clientNameLabel, issuerBankLabel, bankAccountLabel, balanceAmountLabel,

CVCNumberLabel, interestRateLabel, expirationDateLabel

Initialize MyCustomTextField

cardIdTextField, clientNameTextField, issuerBankTextField, bankAccountTextField,  
balanceAmountTextField, CVCNumberTextField, interestRateTextField

Initialize CustomComboBox

expirationDateComboBoxYear ,expirationDateComboBoxMonth,expirationDateComboBoxDay

Initialize MyCustomBtn

addCreditLimitBtn, clearButton ,displayCreditCardBtn

Create a AddCreditCard Constructor and ArrayList Bank Card Parameter for storing BankCards

Set the title name "Add Credit Card "

Set size and default close operation to dispose on Close

Set LocationRelative as null resizable as true and vsibleas true

Create a object add credit card panel

Set the position and size of a panel

Set a titled border with a line border of light blue color and thickness of 2

Set the background color of the panel to dark gray

Set the layout of the panel to null

Add the panel to the frame using the addCreditCardPanel method

Instantiate a MyCustomLabel object with "Card Id" as an argument and assign it to the  
variable "cardIdLabel"

Set the position and size of the cardIdLabel

Add the cardIdLabel to the addCreditCardPanel panel

Instantiate a MyCustomTextField object with an empty string as an argument and assign  
it to the variable cardIdTextField

Set the position and size of the cardIdTextField

Add the cardIdTextField to the addCreditCardPanel panel

Instantiate a MyCustomLabel object with "Client Name" as an argument and assign it to the variable clientNameLabel

Set the position and size of the clientNameLabel

Add the clientNameLabel to the addCreditCardPanel panel

Instantiate a MyCustomTextField object with an empty string as an argument and assign it to the variable clientNameTextField

Set the position and size of the clientNameTextField

Add the clientNameTextField to the addCreditCardPanel panel

Instantiate a MyCustomLabel object with "Issuer Bank" as an argument and assign it to the variable issuerBankLabel

Set the position and size of the issuerBankLabel

Add the issuerBankLabel to the addCreditCardPanel panel

Instantiate a `MyCustomTextField` object with an empty string as an argument and assign it to the variable `issuerBankTextField`

Set the position and size of the `issuerBankTextField`

Add the `issuerBankTextField` to the `addCreditCardPanel` panel

Create a user interface for adding credit card details to a panel called `"addCreditCardPanel"`.

create a label called `"cardIdLabel"` and add it to the panel. Then, create a text field called `"cardIdTextField"` and add it to the panel as well.

create a label called `"clientNameLabel"` and a text field called `"clientNameTextField"`, and add them to the panel.

create a label called `"issuerBankLabel"` and a text field called `"issuerBankTextField"`, and add them to the panel.

create a label called `"bankAccountLabel"` and a text field called `"bankAccountTextField"`, and add them to the panel.

create a label called `"balanceAmountLabel"` and a text field called `"balanceAmountTextField"`, and add them to the panel.

create a label called "CVCNumberLabel" and a text field called "CVCNumberTextField", and add them to the panel.

create a label called "interestRateLabel" and a text field called "interestRateTextField", and add them to the panel.

create a label called "expirationDateLabel" and a combo box called "expirationDateComboBoxYear", and add them to the panel.

Add a CVC number label to the "addCreditCardPanel" panel.

Create a CVC number text field and assign it to a variable called "CVCNumberTextField". Set its position and size, then add it to the "addCreditCardPanel" panel.

Create a custom label called "interestRateLabel" with an argument of "interestRate", and assign it to a variable with the same name. Set its position and size, then add it to the "addCreditCardPanel" panel.

Create a custom text field and assign it to a variable called "interestRateTextField". Set its position and size, then add it to the "addCreditCardPanel" panel.

Create a custom label called "expirationDateLabel" with an argument of "Client Name", and assign it to a variable with the same name. Set its position and size, then add it to the "addCreditCardPanel" panel.

Create a custom combo box and assign it to a variable called "expirationDateComboBoxYear". Set its position and size, but do not add it to the panel yet.

Set the position and size for the "expirationDateComboBoxYear" component

Add the "expirationDateComboBoxYear" component to the "addCreditCardPanel" panel

Instantiate a new "CustomComboBox" object with an empty argument and assign it to the variable "expirationDateComboBoxMonth"

Set the position and size for the "expirationDateComboBoxMonth" component

Add the "expirationDateComboBoxMonth" component to the "addCreditCardPanel" panel

Instantiate a new "CustomComboBox" object with an empty argument and assign it to the variable "expirationDateComboBoxDay"

Set the position and size for the "expirationDateComboBoxDay" component

Add the "expirationDateComboBoxDay" component to the "addCreditCardPanel" panel

Instantiate a new "MyCustomButton" object with the argument "Add Card" and assign it to the variable "addCreditLimitBtn"

Set the position and size for the "addCreditLimitBtn" component

Add the "addCreditLimitBtn" component to the "addCreditCardPanel" panel

Instantiate a new "MyCustomButton" object with the argument "Clear" and assign it to the variable "clearButton"

Set the position and size for the "clearButton" component

Add the "clearButton" component to the "addCreditCardPanel" panel

Instantiate a new "MyCustomButton" object with the argument "Display" (but no assignment or panel addition is specified in the given pseudocode)

Create a CustomComboBox object, set its position and size for the year of expiration date, and add it to the addCreditCardPanel panel.

Create another CustomComboBox object, set its position and size for the month of expiration date, and add it to the addCreditCardPanel panel.

Create a third CustomComboBox object, set its position and size for the day of expiration date, and add it to the addCreditCardPanel panel.

Create a MyCustomButton object with "Add Card" as its argument, assign it to the variable "addCreditLimitBtn", set its position and size, and add it to the addCreditCardPanel panel.

Create a `MyCustomButton` object with "Clear" as its argument, assign it to the variable "clearButton", set its position and size, and add it to the `addCreditCardPanel` panel.

Create a `MyCustomButton` object with "Display" as its argument, assign it to the variable "displayCreditCardBtn", set its position and size, and add it to the `addCreditCardPanel` panel.

When the "Clear" button is clicked, the function `clearEntry()` should be called.

When the "Add Card" button is clicked, the text entered in each of the text fields should be retrieved and assigned to their respective variables. These variables are "cardIdOfCreditCard", "clientNameCreditCard", "issuerBankCreditCard", "bankAccountCreditCard", "balanceAmountCreditCard", "interestRateCreditCard", and "CVCNumberCreditCard".

Set the position and size for the "expirationDateComboBoxYear".

Add the "expirationDateComboBoxYear" to the "addCreditCardPanel" panel.

Instantiate a "CustomComboBox" object with an empty argument and assign it to the variable "expirationDateComboBoxMonth".

Set the position and size for the "expirationDateComboBoxMonth".

Add the "expirationDateComboBoxMonth" to the "addCreditCardPanel" panel.

Instantiate another "CustomComboBox" object with an empty argument and assign it to the variable "expirationDateComboBoxDay".

Set the position and size for the "expirationDateComboBoxDay".

Add the "expirationDateComboBoxDay" to the "addCreditCardPanel" panel.

Instantiate a "MyCustomButton" object with the argument "Add Card" and assign it to the variable "addCreditLimitBtn".

Set the position and size for the "addCreditLimitBtn".

Add the "addCreditLimitBtn" to the "addCreditCardPanel" panel.

Instantiate another "MyCustomButton" object with the argument "Clear" and assign it to the variable "clearButton".

Set the position and size for the "clearButton".

Add the "clearButton" to the "addCreditCardPanel" panel.

Instantiate another "MyCustomButton" object with the argument "Display" and assign it to the variable "displayCreditCardBtn".

Set the position and size for the "displayCreditCardBtn".

Add the "displayCreditCardBtn" to the "addCreditCardPanel" panel.

When the "Clear" button is clicked, call the "clearEntry()" function.

When the "addButton" is clicked, retrieve the text entered in each of the text fields and assign them to their corresponding variables.

If any text field is empty, display the message "Please ensure that all the required fields are completed before proceeding!!".

Else if the "expirationDateComboBox" (year, month, and day) index is 0, display the message "Please kindly select year, month, and day!!".

Else, try to convert the card ID, CVC number, balance amount, and interest rate to their respective data types.

Instantiate a Boolean variable "cardIDFound" and set it to false.

Concatenate the expiration date to a string from the "expirationComboBoxYear", "expirationComboBoxMonth", and "expirationComboBoxDay".

If the card ID is less than or equal to zero, display the message "Invalid card ID. Please enter a positive integer." and return.

Else if the balance amount is less than or equal to zero, display the message "Invalid balance amount. Please enter a positive integer." and return.

Else if the CVC number is less than or equal to zero, display the message "Invalid CvcNumber. Please enter a positive integer." and return.



Else if the interest rate is less than or equal to zero, display the message "Invalid interestRate. Please enter a positive integer." and return.

Check whether the card ID already exists. If "cardIDFound" is true, display the message "Card identification number has already been used or claimed by someone else!!".

Else, instantiate a "CreditCard" object using "cardId, clientName, issuerBankCreditCard, bankAccountCreditCard, balanceAmount, cvcNumber, interestRate, expirationDate" as arguments and assign it to the variable "creditCard".

Add the "creditCard" object to "bankcards".

Display the message "Credit Card added successfully".

Call the "clearEntry()" method.

Catch the "NumberFormatException".

Set the position and size for the expiration date combo box year, and add it to the "addCreditCardPanel" panel.

Instantiate a CustomComboBox object and assign it to the variable "expirationDateComboBoxMonth", set its position and size, and add it to the "addCreditCardPanel" panel.

Instantiate another CustomComboBox object and assign it to the variable "expirationDateComboBoxDay", set its position and size, and add it to the "addCreditCardPanel" panel.

Instantiate a MyCustomButton object with the label "Add Card" and assign it to the variable "addCreditLimitBtn", set its position and size, and add it to the "addCreditCardPanel" panel.

Instantiate another MyCustomButton object with the label "Clear" and assign it to the variable "clearButton", set its position and size, and add it to the "addCreditCardPanel" panel.

Instantiate a third `MyCustomButton` object with the label "Display" and assign it to the variable "displayCreditCardBtn", set its position and size, and add it to the "addCreditCardPanel" panel.

When the "Clear" button is clicked, call the "clearEntry()" function.

When the "Add Card" button is clicked, retrieve the values entered in each of the text fields, convert them to the appropriate data type, and assign them to their respective variables.

Check if any of the text fields are empty. If so, display a message asking the user to fill in all the required fields.

Check if the expiration date combo box has been fully selected. If not, display a message asking the user to select the year, month, and day.

If all input is valid, instantiate a `CreditCard` object with the entered values, add it to the "bankcards" collection, and display a success message. If the card ID already exists, display an error message.

When the "Display" button is clicked, create a new `DisplayCreditCard` object with the "bankcards" collection as an argument.

Create a "clearEntry()" method

Define a method named "clearEntry" that clears all the input fields after the "Add Card" or "Clear" button is clicked. This method will clear the text field for the card ID, client name, issue bank, bank account, balance amount, CVC number, interest rate, and expiration date.

Clear the text field for the card ID:

In the "clearEntry" method, clear the text field for the card ID by setting its value to an empty string.

Clear the text field for the client name:

In the "clearEntry" method, clear the text field for the client name by setting its value to an empty string.

Clear the text field for the issue bank:

In the "clearEntry" method, clear the text field for the issue bank by setting its value to an empty string.

Clear the text field for the bank account:

In the "clearEntry" method, clear the text field for the bank account by setting its value to an empty string.

Clear the text field for the balance amount:

In the "clearEntry" method, clear the text field for the balance amount by setting its value to an empty string.

Clear the text field for the CVC number:

In the "clearEntry" method, clear the text field for the CVC number by setting its value to an empty string.

Clear the text field for the interest rate:

In the "clearEntry" method, clear the text field for the interest rate by setting its value to an empty string.

Clear the text field for the expiration date:

In the "clearEntry" method, clear the text field for the expiration date by setting its value to an empty string.

End the "clear" method:

After all the text fields have been cleared in the "clearEntry" method, the method ends.

### 3.9 Pseudocode For Add Debit Card

Add addDebitCardBtn to addDebitCardPanel panel

Instantiate a CustomComboBox object, using the ArrayList of BankCard objects passed in as an argument and assign it to the variable "issueBankComboBoxDebitCard".

Set Position and size for issueBankComboBoxDebitCard

Add issueBankComboBoxDebitCard to addDebitCardPanel panel

Instantiate a MyCustomLabel object, using "Interest Rate" as an argument and assign it to the variable "interestRateLabelDebitCard".

Set Position and size for interestRateLabelDebitCard

Add interestRateLabelDebitCard to addDebitCardPanel panel

Instantiate a MyCustomTextField object, using "" as an argument and assign it to variable interestRateTextFieldDebitCard

Set Position and size for interestRateTextFieldDebitCard

Add interestRateTextFieldDebitCard to addDebitCardPanel panel

Instantiate a MyCustomLabel object, using "CVC" as an argument and assign it to the variable "cvcLabelDebitCard".

Set Position and size for cvcLabelDebitCard

Add cvcLabelDebitCard to addDebitCardPanel panel

Instantiate a MyCustomTextField object, using "" as an argument and

assign it to variable `cvcTextFieldDebitCard`

Set Position and size for `cvcTextFieldDebitCard`

Add `cvcTextFieldDebitCard` to `addDebitCardPanel` panel

Create a method `clearEntry`

Clear the text field for the card ID.

Clear the text field for the client name.

Clear the text field for the issue bank.

Clear the text field for the bank account.

Clear the text field for the balance amount.

Clear the text field for the CVC number.

Clear the text field for the interest rate.

End the "clear" method.

Create a method `addCard`

Validate that all input fields are in correct format (card ID, balance amount, CVC number, and interest rate).

If any field is invalid, display "Invalid input format. Please enter a valid number for card ID, balance amount, CVC number, and interest rate".

Else, create a new `DebitCard` object with the values from the input fields.

Add the new `DebitCard` object to the `ArrayList` of `BankCard` objects.

Display a message "Card Added Successfully".

Call the `clearEntry` method to clear the input fields.

End the "addCard" method.

Add addDebitCardBtn to addDebitCardPanel panel

Instantiate a MyCustomButton object, using "Clear" as an argument and assign it to the variable "clearBtnDebitCard".

Set Position and Size for clearBtnDebitCard

Add clearBtnDebitCard to addDebitCardPanel panel

Instantiate a MyCustomButton object, using "Display" as an argument and assign it to the variable "displayBtnDebitCard".

Set Position and Size for displayBtnDebitCard

Add displayBtnDebitCard to addDebitCardPanel panel

Instantiate a MyCustomTextField object, using "" as an argument and assign it to variable cvcTextFieldDebitCard

Set Position and size for cvcTextFieldDebitCard

Add cvcTextFieldDebitCard to addDebitCardPanel panel

Instantiate a MyCustomLabel object, using "CVC Number" as an argument and assign it to the variable "cvclabelDebitCard".

Set Position and size for cvclabelDebitCard

Add cvclabelDebitCard to addDebitCardPanel panel

Instantiate a MyCustomTextField object,using "" as an argument and assign it to variable interestRateTextFieldDebitCard

Set Position and size for interestRateTextFieldDebitCard

Add interestRateTextFieldDebitCard to addDebitCardPanel panel

Instantiate a MyCustomLabel object, using "Interest Rate" as an argument and assign it to the variable "interestRateLabelDebitCard".

Set Position and size for interestRateLabelDebitCard

Add interestRateLabelDebitCard to addDebitCardPanel panel

Instantiate a MyCustomTextField object,using "" as an argument and assign it to variable expiryDateTextFieldDebitCard

Set Position and size for expiryDateTextFieldDebitCard

Add expiryDateTextFieldDebitCard to addDebitCardPanel panel

Instantiate a MyCustomLabel object, using "Expiry Date" as an argument and assign it to the variable "expiryDateLabelDebitCard".

Set Position and size for expiryDateLabelDebitCard

Add expiryDateLabelDebitCard to addDebitCardPanel panel

Instantiate a CustomComboBox object with bankcards as argument and assign it to the variable bankComboBoxDebitCard

Set Position and size for bankComboBoxDebitCard



Add bankComboBoxDebitCard to addDebitCardPanel panel

Create method clearEntry

Clear the text field for the card ID.

Clear the text field for the client name.

Clear the text field for the issue bank.

Clear the text field for the bank account.

Clear the text field for the balance amount.

Clear the text field for the CVC number.

Clear the text field for the interest rate.

Clear the text field for the expiration date.

End the "clear" method.

Create method displayCreditCard

Declare variable cardId as String and assign it the value of the

getText of the cardIdTextFieldDebitCard.

Declare variable clientName as String and assign it the value of

the getText of the clientNameTextFieldDebitCard.

Declare variable bankAccount as String and assign it the value of

the getText of the bankAccountTextFieldDebitCard.

Declare variable balance as double and assign it the value of the

getText of the balanceAmountTextFieldDebitCard converted to double.

Declare variable cvc as int and assign it the value of the getText of

the `cvcTextFieldDebitCard` converted to `int`.

Declare variable `interestRate` as `double` and assign it the value of the

`getText` of the `interestRateTextFieldDebitCard` converted to `double`.

Declare variable `expiryDate` as `String` and assign it the value of the

`getText` of the `expiryDateTextFieldDebitCard`.

Declare variable `bank`

Add the `"addDebitCardBtn"` button to the `"addDebitCardPanel"` panel.

Create a button object `"clearBtnDebitCard"` using `"Clear"` as an argument and add it to the panel.

Create a button object `"displayBtnDebitCard"` using `"Display"` as an argument and add it to the panel.

When the `"addButton"` is clicked, retrieve the text entered in each text field and assign them to their respective variables.

If any text field is empty, display a message asking to complete all required fields.

Otherwise, try to convert the entered values into the appropriate data types and perform necessary checks.

If the card ID is invalid or has already been used, display an appropriate message.

Otherwise, create a new `"debitCard"` object and add it to the `"bankcards"` list.

Display a message confirming the successful addition of the debit card and clear all text fields using the `"clearEntry()"` method.

When the `"Display"` button is clicked, create a new `"DisplayDebitCard"` object with the `"bankcards"` list as an argument.

When the `"Clear"` button is clicked, call the `"clearEntry()"` function to clear all text fields.

The `"clearEntry()"` function clears all the text fields in the form.

### 3.10 Pseudo Codes for CancelCreditCard

START

Create a class CancelCreditCard extends JFrame

Declare variables for customTextField

Declare variables for customLabel

Declare variables for customButton

Create Constructor cancelCreditCard with BankCard ArrayList

setTitle Cancel Credit Card

Set Default Close Operation(Dispose ON Close)

Set Size

Set Location

Set visible to True

Create JPanel for cancelCreditCard

Set Size and coordinates

Set Layout to null

Create JLabel for Cancel Credit Card

Set Foreground

Set Font

Set size and coordinates

Add Label to cancelCreditCardPanel

Add Panel to the frame

Create label for CardID

Set Size and Coordinates

Add label to cancelCreditCardPanel

Create text field for CardID

Set Size and Coordinates

Add text field to cancelCreditCardPanel

Create Done Button

Set Size and coordinates

Add button to cancelCreditCardPanel

Create Clear Button

Set Size and coordinates

```
Set Size and coordinates

When done button is pressed

Call clearEntry function

When clear button is pressed

Assign creditCardCardId to cardId

IF cardId is empty THEN

Display Enter a Card ID

ELSE

TRY

Assign cardId to newCardId

Use FOR loop to iterate through BankCard ArrayList for the

Details to cast CreditCard

IF casting is successful THEN

IF getter method from BankCard cardId is equal to

newCardId THEN

Call cancelCreditCard method from CreditCard

Display Credit Card successfully cancelled.

Call clearEntry function

Return

END IF

END
```

### 3.11 Pseudo Code for DisplayCreditCard

Declare package CourseWorkTwo.views

Import BankCard and CreditCard classes from CourseWorkTwo package

Import java.swing.\*

Import java.awt.\*

import java.util.ArrayList

Create a class DisplayCreditCard which extends JFrame

Create a constructor DisplayCreditCard with an ArrayList BankCard

parameter to store bankcards

Set title "Display Credit Cards"

Set size of frame

Set default close operation to Dispose on close

Set Location Relative to null

Set resizable to false

Set visible to true

Create a displaypanel

Set layout of panel to BoxLayout in Y\_AXIS

Create a 2D String array named data with the size of bankCards.size()

and 9."

To iterate through each element in the bankCards collection using a for

loop, attempt to cast each element to a Credit Card object. If the

casting

is successful, use the get methods of the Credit Card object to set the

corresponding values in a data array.

Create a new JTable object named table with data and columnNames as its parameters.

Create a new JScrollPane object named scrollPane with table as its parameter

Add scrollPane to panel

### 3.12 Pseudo Codes for DisplayDebitCard

START

Create Class DisplayDebitCard extends JFrame

Create constructor DisplayDebitCard with BankCard ArrayList

Set Title Display Debit Card

Set DefaultCloseOperation to Dispose On Close

Set Size

Set Location Relative to null

Set visible to True

Create JPanel for debitCardDisplayPanel

Set layout of debitCardDisplayPanel

Create 2D array with headerName

Use LOOP to iterate through each element in bankCardOne

To cast each element to DebitCard object

IF casting is successful THEN

Use getter method of DebitCard to set the

corresponding value in array

Create JTable named debitCardDisplayTable with details and

headerName as parameters

Create scrollPane with debitCardDisplayTable as parameter

Add scrollPane to the panel

Add panel to the frame

Add addDebitCardBtn to addDebitCardPanel panel

Instantiate a MyCustomButton object, using "Clear" as an argument

and assign it to the variable "clearBtnDebitCard".

Set Position and Size for clearBtnDebitCard

Add clearBtnDebitCard to addDebitCardPanel panel



Instantiate a MyCustomButton object, using "Display" as an argument and assign it to the variable "displayBtnDebitCard".

Set Position and Size for displayBtnDebitCard

Add displayBtnDebitCard to addDebitCardPanel panel

When the "addButton" is clicked, retrieve the text entered in each of the text fields.

Assign the value of the text entered in the

"balanceAmountTextFieldDebitCard" to the variable named

"balanceAmountOfDebitCard".

Assign the value of the text entered in the

"cardIdTextFieldDebitCard" to the variable named

"cardIDOfDebitCard".

Assign the value of the text entered in the

"bankAccountTextFieldDebitCard" to the variable named

"bankAccountOfDebitCard".

Assign the value of the text entered in the

"issuerBankTextFieldDebitCard" to the variable named

"issuerBankOfDebitCard".

Assign the value of the text entered in the

"clientNameTextFieldDebitCard" to the variable named

"clientNameOfDebitCard".

Assign the value of the text entered in the

"pinNumberTextFieldDebitCard" to the variable named

"pinNumberOfDebitCard".

IF any text field is empty THEN

It displays " Please ensure that all the required fields are  
completed before proceeding!!"

ELSE

TRY

Convert balaneAmountOfDebitCard to double

Convert CardIDOfDebitCard to int

Convert pinNumberOfDebitCard to int

Instantiate Boolean cardIDFound to False

IF CardID less than or equal to zero

Displays "Invalid card ID. Please enter a  
positive integer."

ELSE IF balanceAmount less than or equal to zero

Displays "Invalid balance amount. Please  
enter a positive integer."

ELSE IF pinNumber less than or equal to zero

Displays "Invalid pinNumber. Please enter a

positive integer."

ELSE checks whether the card id already exists

IF cardIdFound is true Then

Displays" Card identification number has

already been used or claimed by someone

else!!"

ELSE

Instantiate a CreditCard object, using "

balanceAmount, cardId, bankAccountOfDebitCard,

issuerBankOfDebitCard, clientNameOfDebitCard,

pinNumber" as an argument and assign it to the

variable "debitCard".

Add debitCard to bankcards

Displays" Debit Card added successfully"

Call clearEntry() method

CATCH numberformatexception

Displays" Invalid input format. Please enter a valid

number for balance amount, card ID, and PIN number."

When the "Display" is clicked

Create a new DisplayDebitCard object with bankcards as an argument

When the "Clear" is clicked

Call function clearEntry()

Create method clearEntry

Clear the text field for the card ID.

Clear the text field for the client name.

Clear the text field for the bank account

Clear the text field for the issue bank.

Clear the text field for the balance amount.

Clear the text field for the pin number

End the "clear" method. parapharis this code

### 3.13 Pseudocode For Bank GUI

Import the required packages and classes for the application.

Define a new class called BankGUI that extends JFrame.

Declare an ArrayList of BankCard objects called bankCards.

Declare a set of JPanels, MyCustomButtons, and MyCustomLabels.

Create a constructor for the BankGUI class.

Set the background color of the JFrame to white.

Set the size of the JFrame.

Set the location of the JFrame relative to null.

Set the visibility of the JFrame to true.

Set the layout of the JFrame to null.

Set the default close operation of the JFrame to exit on close.

Create a main panel.

Set the size and position of the main panel.

Set the border of the main panel.

Set the background color of the main panel.

Set the size and position of a JLabel called bankLabel.

Set the font of the bankLabel to Arial size 20.

Set the foreground color of the bankLabel to white.

Set the horizontal and vertical alignment of the bankLabel to center.

Instantiate a MyCustomButton object called addDebitCardBtn with the argument "Add Debit Card" and set its position and size.

Instantiate a MyCustomButton object called addCreditCardBtn with the argument "Add Credit Card" and set its position and size.

Instantiate a MyCustomButton object called withdrawalBtn with the argument "Withdrawal From Debit Card" and set its position and size.

Instantiate a MyCustomButton object called setCreditLimitBtn with the argument "Pseudo Code for AddDebitCard" and set its position and size.

Instantiate a MyCustomButton object called cancelCreditCardBtn with the argument "Cancel Credit Card" and set its position and size.

Instantiate a MyCustomButton object called displayBtn with the argument "Display" and set its position and size.

Add an action listener to addDebitCardBtn that creates an instance of the AddDebitCard class, passing the bankCards object as a parameter to its constructor.

Add an action listener to addCreditCardBtn that creates an instance of the AddCreditCard class, passing the bankCards object as a parameter to its constructor.

Add an action listener to withdrawalBtn that creates an instance of the WithdrawFromDebitCard class, passing the bankCards object as a parameter to its constructor.

Add an action listener to setCreditLimitBtn that creates an instance of the SetCreditLimit class, passing the bankCards object as a parameter to its constructor.

Add an action listener to cancelCreditCardBtn that creates an instance of the CancelCreditCard class, passing the bankCards object as a parameter to its constructor.

Add an action listener to displayBtn that creates an instance of the MainDisplay class, passing the bankCards object as a parameter to its constructor.

Add the bankLabel, addCreditCardBtn, addDebitCardBtn, withdrawalBtn, setCreditLimitBtn, displayBtn, and main panel to the JFrame.

Create the main method.

Call the constructor of the BankGUI class.

### 3.14 Pseudocode for mainDisplay

Import the following classes: CourseWorkTwo.BankCard, java.util.ArrayList, javax.swing.\*

Create a class called MainDisplay that extends JFrame

Declare a variable called table of type JTable

Create a constructor for MainDisplay that takes an ArrayList of BankCard objects as a parameter:

Call the parent constructor and pass the string "Display" as an argument

Set the default close operation to JFrame.Hide on close

Set the size of the frame to 500 x 500 pixels

Set the location of the frame to the center of the screen

Set visible to true

Create a new JPanel called panel

Define an array of strings named columnNames that holds the column headers for a table

Create a 2D string array called data with the same number of rows as the size of the bankCards ArrayList, and 5 columns

Use a for loop to iterate through each BankCard object in the bankCards ArrayList:

Set the value in the data array at row i and column 0 to the card ID of the BankCard object

Set the value in the data array at row i and column 1 to the bank account of the BankCard object

Set the value in the data array at row i and column 2 to the balance amount of the BankCard object

Set the value in the data array at row i and column 3 to the client name of the BankCard object

Set the value in the data array at row i and column 4 to the card type of the BankCard object

Create a new JTable object called table, passing in the data array and columnNames array as arguments

Create a new JScrollPane object called scrollPane, passing in the table object as an argument

Add the scrollPane object to the panel object



Add the panel object to the JFrame object

### 3.15 Pseudo Code For SetCreditLimit

Declare package CourseWorkTwo.views

Import BankCard and CreditCard classes from courseWork2 package

Import CustomComboBox, MyCustomButton, MyCustomLabel, MyCustomTextField  
from courseWork2.components package

Import java.swing.\*

Import java.awt.\*

Create a class AddCreditCard which extends JFrame

Declare

MyCustomLabel, MyCustomTextField, CustomComboBox, CustomButton  
variables

Create a constructor SetCreditLimit with an ArrayList BankCard parameter to  
store bankcards

Set title "Set Credit Limit"

Set size to the frame

Set default close operation to Dispose on close

Set Location Relative to null

Set resizable to false

Set visible to true

Create a setCreditLimitPanel

Set Position and Size of panel

Set titled Border along with line border with lightblue color and 2 thickness

Set Background to red

Set Layout to null

Add setCreditLimitPanel to frame

Instantiate a MyCustomLabel object, using "Card Id" as an argument and assign it to the variable "cardIdLabelSetCreditLimit".

Set Position and size for cardIdLabelSetCreditLimit

Add cardIdLabelSetCreditLimit to setCreditLimitPanel panel

Instantiate a MyCustomTextField object,using "" as an argument and assign it to variable cardIdTextFieldSetCreditLimit

Set Position and size for cardIdTextFieldSetCreditLimit

Add cardIdTextFieldSetCreditLimit to setCreditLimitPanel panel

Instantiate a MyCustomLabel object, using "Credit Limit" as an argument and assign it to the variable "creditLimitLabel".

Set Position and size for CreditLimitLabel

Add creditLimitlabel to setCreditLimitPanel panel

Instantiate a MyCustomTextField object,using "" as an argument and assign it to variable creditLimitTextField

Set Position and size for creditLimitTextField

Add creditLimitTextField to setCreditLimitPanel panel

Instantiate a MyCustomLabel object, using "Credit Limit" as an argument and assign it to the variable

"gracePeriodSetCreditLimitLabel".

Set Position and size for gracePeriodSetCreditLimitLabel

Add gracePeriodSetCreditLimitLabel to setCreditLimitPanel panel

Instantiate a MyCustomTextField object,using "" as an argument and assign it to variable gracePeriodSetCreditLimitTextField

Set Position and size for gracePeriodSetCreditLimitTextField

Add gracePeriodSetCreditLimitTextField to setCreditLimitPanel panel

Instantiate a MyCustomButton object, using "Set Limit" as an argument and assign it to the variable "setCreditLimitBtn".

Set Position and Size for setCreditLimitBtn

Add setCreditLimitBtn to setCreditLimitPanel panel

Instantiate a MyCustomButton object, using "Clear" as an argument and assign it to the variable "setCreditLimitClearBtn".

Set Position and Size for setCreditLimitClearBtn

Add setCreditLimitClearBtn to setCreditLimitPanel panel

When the "Clear" is clicked

Call clearEntry method

When the "SetLimit" is clicked, retrieve the text entered in

cardID,creditLimit,gracePeriod

TRY

Assign the value of the text entered in the

"cardIdTextFieldSetCreditLimit" to the variable named "cardId".

Assign the value of the text entered in the "creditLimitTextField"

to the variable named "creditLimit".

Assign the value of the text entered in the

"gracePeriodSetCreditLimitTextField" to the variable named

"gracePeriod".

IF any text field is empty THEN

It displays" Please ensure that all the required fields are

completed before proceeding!!"

RETURN

IF newCardId,newCreditLimit,newGracePeriod is less than or

equal to zero

THEN

Displays” Invalid input detected for the card ID,  
credit limit, and grace period fields!!”

RETURN

IF creditCard is null or already exists

THEN

Displays” The card you entered is not recognized  
as a credit card!”

RETURN

Checks IF newCreditLimit is greater than 2.5 times of  
getBalanceAmount from Credit Card

THEN

Displays” The credit card cannot be issued more  
credit than 2.5 times its current balance, and the  
requested amount exceeds this limit!

Pass newCreditLimit,newGracePeriod as an argument in  
setCreditLimit() method from Credit Card

Displays The system has updated your credit limit  
successfully:

Calls clearEntry method

CATCH numberFormatException

Displays " Invalid input detected. Please input only

valid numbers for the card ID, credit limit, and

grace period fields.!!to the variable named

"balanceAmount".

Create a method to find a CreditCard with a given card ID from a list of BankCards

Iterate through the list of BankCards

Check IF the current BankCard is a CreditCard and if it has the given

card ID

IF found

Return the CreditCard

otherwise

return null

Create method clearEntry

Clear the text field for the card ID.

Clear the text field for the creditLimit.

Clear the text field for the gracePeriod.

End the "clear" method.

### 3.16 Pseudo Code for Withdraw From Debit Card

Declare package CourseWorkTwo.views

Import BankCard and CreditCard classes from CourseWorkTwo package

Import

CustomComboBox,MyCustomButton,MyCustomLabel,MyCustomTextField,CustomC

omboBox

from CourseWorkTwo.components package

Import java.swing.\*

Import java.awt.\*

Create a class AddCreditCard which extends JFrame

Declare MyCustomLabel,MyCustomTextField,CustomComboBox,CustomButton

variables

Create a constructor WithdrawFromDebitCard with and ArrayList BankCard

parameter to store bankcards

Set title "Withdraw from debit card"

Set size to the frame

Set default close operation to Dispose on close

Set Location Relative to null

Set resizable to false

Set visible to true

Create a withdrawalPanel

Set Position and Size of panel

Set titled Border along with line border with lightblue color and 2 thickness

Set Background to red

Set Layout to null

Add setCreditLimitPanel to frame

Instantiate a MyCustomLabel object, using "Card Id" as an argument and assign it to the variable "cardIdLabelWithdrawal".

Set Position and size for cardIdLabelWithdrawal

Add cardIdLabelWithdrawal to withdrawalPanel panel

Instantiate a MyCustomTextField object, using "" as an argument and assign it to variable cardIdTextFieldWithdrawal

Set Position and size for cardIdTextFieldWithdrawal

Add cardIdTextFieldWithdrawal to withdrawalPanel panel

Instantiate a MyCustomLabel object, using "Withdraw Amount" as an argument and assign it to the variable "withdrawalAmountLabel".

Set Position and size for withdrawalAmountLabel

Add withdrawalAmountLabel to withdrawalPanel panel



Instantiate a MyCustomTextField object,using "" as an argument and assign it to

variable withdrawalAmountTextField

Set Position and size for withdrawalAmountTextField

Add withdrawalAmountTextField to withdrawalPanel panel

Instantiate a MyCustomLabel object, using "Date Of Withdraw" as an argument

and assign it to the variable "dateOfWithdrawalLabel".

Set Position and size for dateOfWithdrawalLabel

Add dateOfWithdrawalLabel to withdrawalPanel panel

Instantiate a CustomComboBox object,using String Array which takes years values

and assign it to variable withdrawalComboBoxYear

Set Position and size for withdrawalComboBoxYear

Add withdrawalComboBoxYear to withdrawalPanel panel

Instantiate a CustomComboBox object,using String Array which takes months values

and assign it to variable withdrawalComboBoxMonth

Set Position and size for withdrawalComboBoxMonth

Add withdrawalComboBoxMonth to withdrawalPanel panel

Instantiate a CustomComboBox object,using String Array which takes days values

and assign it to variable withdrawalComboBoxDay

Set Position and size for withdrawalComboBoxDay

Add withdrawalComboBoxDay to withdrawalPanel panel

Instantiate a MyCustomLabel object, using "Date Of Withdraw" as an argument and assign it to the variable "pinNumberWithdrawalLabel".

Set Position and size for pinNumberWithdrawalLabel

Add pinNumberWithdrawalLabel to withdrawalPanel panel

Instantiate a MyCustomTextField object, using "" as an argument and assign it to variable pinNumberWithdrawalTextField

Set Position and size for pinNumberWithdrawalTextField

Add pinNumberWithdrawalTextField to withdrawalPanel panel

Instantiate a MyCustomButton object, using "Withdraw" as an argument and assign it to the variable "withdrawalBtn".

Set Position and Size for withdrawalBtn

Add withdrawalBtn to withdrawalPanel panel

Instantiate a MyCustomButton object, using "Clear" as an argument and assign it to the variable "clearBtnWithdrawal".

Set Position and Size for clearBtnWithdrawal

Add clearBtnWithdrawal to withdrawalPanel panel

Instantiate a MyCustomButton object, using "Display" as an argument and assign it

to the variable "displayBtn".

Set Position and Size for displayBtn

Add displayBtn to withdrawalPanel panel

When the "Display" is clicked

Create a new DisplayDebitCard object with bankcards as an argument

When the "Clear" is clicked

Call function clearEntry()

When "Withdraw" is clicked get all the values from card Id, withdrawal Amount, date of

Withdrawal, pin number

Checks IF any field is Empty

Displays "Please ensure that all the required fields are completed  
before proceeding!!"

ELSE

TRY

Parse the value of cardId to int as withdrawalCardId variable

Parse the value of withdrawalAmount to int as withdrawal  
variable

Parse the value of pinNumber to int as withdrawalPin variable

Instantiate cardFound as Boolean to false and debitCard to null

IF withdrawal CardId is less than or equal to zero

Displays "Invalid card Id! Please enter a positive integer"

ELSE

Iterate through the list of BankCards

Check IF the current BankCard is a DebitCard and if it has the  
given card ID

IF getCardId from debitCard matches withdrawalCardId

Cardfound is true

BREAK

IF not card Found

Displays "The card you entered cannot be found in the system."

ELSE IF withdrawal is less than or equal to zero

Displays " Invalid withdrawal amount. Please enter a positive  
number."

ELSE

Pass debitcard,withdrawalPin,withdrawal,dateOfwithdrawal as  
argument to withdraw method

CATCH exception

Displays " Invalid input format. Please enter a valid number for c  
card ID, withdrawal balance amount, and PIN number!"

Create method withdraw passing argument as debitcard ,pin,withdraw

,dateofwithdraw

IF getPinNumber from debitCard matches pin

IF debit card getBalanceAmount is less than withdrawal

Displays” The account does not have sufficient funds to process

the requested transaction!!”

RETURN

Pass argument withdrawal,dateOFwithdrawal,pin to withdraw from

debitcard

Displays”Your account has been successfully debited with the

withdrawn amount :)”

Call ClearEntry method

ELSE

Displays”The pin you entered is incorrect”

Create a method clearEntry

Clear the text field for the card Id

Clear the text field for the withdrawal amount

Clear the text field for the pin Number

Set the value of withdrawalComboBoxYear to index 0

Set the value of withdrawalComboBoxMonth to index 0

Set the value of withdrawalComboBoxDay to index 0

## 4. METHOD DESCRIPTION

### 4.1 Method description do CustomComboBox

This Java class 'CustomComboBox' extends the 'JComboBox' class and provides a custom implementation for a combo box GUI component.

The constructor of this class takes an array of strings as input, which are the values to be displayed in the combo box. These values are passed to the parent 'JComboBox' class constructor.

The 'setFont' method is called in the constructor to set the font style of the text displayed in the combo box. Here, the font is set to "Tohama", bold and a size of 15.

The `setMaximumRowCount` method is also called in the constructor, which sets the maximum number of items to be displayed in the combo box drop-down list. In this case, it is set to 5.

Overall, this `CustomComboBox` class allows for customization of the font style and maximum item count of a combo box component, providing a more flexible implementation for developers.

#### Method Description for MyCustomButton

This is a class called `MyCustomBtn` that extends the `JButton` class in Java Swing. It defines a customized button that can be used in a graphical user interface (GUI) application.

The class has a constructor that takes a `String` argument, which is the text that will be displayed on the button. Within the constructor, the `super` keyword is used to call the constructor of the parent `JButton` class, passing in the text argument.

The `setFocusPainted(false)` method is called to disable the button's focus painting, which removes the border around the button when it is clicked. The `setContentAreaFilled(true)` method is called to make sure the button is opaque.

The `setFont` method sets the font of the button to "Dialog" with a bold weight and a size of 15 points. The `setForeground` method sets the text color to white, and the `setBackground` method sets the background color to a dark greenish-blue (RGB value 0,102,102).

Finally, the `setMargin` method is used to set the insets of the button. In this case, there is a margin of 10 pixels on all sides of the button, which provides some spacing between the button's text and its border.

Overall, this class provides a customizable button that can be used in a Java Swing GUI application with specific visual characteristics.

## 4.2 Method Description For MyCustomLabel

This is a Java class called `MyCustomLabel` that extends the `JLabel` class from the `javax.swing` package. It provides a custom label component that has a constructor that takes in a `String` argument which represents the text that will be displayed on the label.

In the constructor, the `super()` method is called to pass the `text` parameter to the parent class. The `setFont()` method is called to set the font of the label to "Tohama" with a font size of 15 and the `setForeground()` method is called to set the text color to white.

This custom label component can be used to create labels with a specific font and color in a Java Swing user interface.

## 4.3 Method Description For My Custom Text Field

This Java class defines a custom text field component that extends the basic `JTextField` class in the Swing framework. The `MyCustomTextField` class adds some custom functionality and styling to the text field component.



The class has a constructor that takes a `String` parameter for the initial text that is displayed in the text field. It calls the constructor of the `JTextField` superclass with this parameter to initialize the text field.

In the constructor, the `setColumns` method is called to set the preferred width of the text field in columns. The `setBorder` method is also called to set a custom line border with a gray color and a thickness of 2 pixels.

Finally, the `setFont` method is called to set the font of the text field to a bold Arial font with a size of 14 points.

This custom text field component can be used in Java Swing applications by creating an instance of the `MyCustomTextField` class and adding it to a container using the appropriate layout manager.

#### 4.4 Method Description For AddCreditCard

This code appears to be a Java Swing user interface for adding a credit card. The interface consists of various components such as text fields, labels, and combo boxes, which are used to collect information about the credit card being added.

The `AddCreditCard` class extends the `JFrame` class, which provides a window for displaying the user interface. The constructor takes an `ArrayList` of `BankCard` objects as a parameter.

The JPanel class is used to create a panel for the credit card form, which is given a titled border using the TitledBorder and LineBorder classes. The panel is also given a background color and a null layout. The various components such as labels, text fields, and combo boxes are then added to the panel using the add method.

The MyCustomLabel, MyCustomTextField, and CustomComboBox classes appear to be custom components created for the application. They likely extend the standard Swing components to provide custom functionality and styling.

Overall, this code appears to be a part of a larger application for managing bank accounts and credit cards. It provides a user interface for adding credit cards to the system.

#### 4.5 Method Description For AddDebitCard

This is a Java class `AddDebitCard` which extends `JFrame` to create a graphical user interface (GUI) for adding a new debit card. It imports the `BankCard` and `DebitCard` classes from another package, as well as custom Swing components from the `Components` package, which likely extend the standard Swing components with additional functionality.

The class initializes and sets the properties of the JFrame, and creates a JPanel to hold the components of the GUI. It then creates several custom Swing components such as `MyCustomLabel`, `MyCustomTextField`, and `MyCustomButton` for displaying labels, text fields, and buttons respectively.

Each debit card field such as `cardIdTextFieldDebitCard` and `balanceAmountTextFieldDebitCard` is associated with a corresponding label such as `cardIdLabelDebitCard` and

``balanceAmountLabelDebitCard``. The GUI also includes an "Add Debit Card" button, a "Clear" button, and a "Display" button.

The constructor takes an ``ArrayList`` of ``BankCard`` objects as an argument, presumably to add the new debit card to the list of existing bank cards.

Overall, this code appears to be part of a larger application for managing bank cards, with this class specifically handling the addition of new debit cards.

## 4.6 Method Description For Cancel Credit Card

This is a Java Swing GUI application that allows users to cancel a credit card. The ``CancelCreditCard`` class extends the ``JFrame`` class, which is a top-level container that holds all the components in the GUI application. The class contains three fields: ``cancelCreditCardIdLabel``, ``cancelCreditCardCardIdTextField``, and ``cancelCreditCardBtn``, which are the label, text field, and button components respectively.

The ``CancelCreditCard`` constructor takes an ``ArrayList`` of ``BankCard`` objects as its parameter. This parameter is used to check if the entered card ID exists in the bankCards list. The constructor sets up the

window by calling various methods such as `setSize()`, `setDefaultCloseOperation()`, `setLocationRelativeTo()`, `setResizable()`, and `setVisible()`.

The `CancelCreditCard` constructor also creates a `JPanel` object called `cancelCreditCardPanel` and adds it to the frame. This panel is used to group the components related to canceling a credit card. The panel is given a `TitledBorder`, `LineBorder`, and a background color to create a distinct look.

The `cancelCreditCardIdLabel` and `cancelCreditCardCardIdTextField` components are added to the `cancelCreditCardPanel`. The label component displays the text "Card ID", while the text field component is used to enter the card ID to be canceled. The `cancelCreditCardBtn` and `cancelCreditCardClearBtn` components are also added to the `cancelCreditCardPanel`. The `cancelCreditCardBtn` is the button component that triggers the cancellation of a credit card, while the `cancelCreditCardClearBtn` is the button component that clears the input fields.

The `cancelCreditCardClearBtn` and `cancelCreditCardBtn` buttons have event listeners that respond to button clicks. The `cancelCreditCardClearBtn` listener calls the `clearEntry()` method, which clears the text field component. The `cancelCreditCardBtn` listener gets the card ID entered in the text field component and checks if it exists in the `bankCards` list. If it exists, the credit card with that ID is canceled by calling the `cancelCreditCard()` method of the `CreditCard` object. If it doesn't exist, an error message is displayed.

Overall, this application provides a simple interface for canceling credit cards, and it demonstrates how to use various Swing components and event listeners to create a functional GUI application.

## 4.7 Method Description For Display Credit Card

This code defines a `DisplayCreditCard` class that extends the `JFrame` class and is used to display the details of all credit cards in the input `ArrayList<BankCard> bankCards`.

The constructor of the `DisplayCreditCard` class first calls the constructor of the `JFrame` class with the title "Display Credit Cards", sets the default close operation to `HIDE_ON_CLOSE`, sets the size of the frame to 800x700 pixels, centers the frame on the screen, disables resizing of the frame, and makes the frame visible.

Then, a `JPanel` object called `displayPanel` is created with a vertical `BoxLayout`. The column headers of the table are stored in a `String` array called `headersName`.

Next, an `ArrayList` of `CreditCard` objects called `creditCards` is created by iterating over the input `bankCards` and adding all `CreditCard` objects to the list. A 2D `String` array called `detail` is also created with the same number of rows as the size of `creditCards` and 9 columns to hold the details of each credit card.

The details of each credit card in the `creditCards` list are then added to the `detail` array by accessing their properties using getter methods. Finally, a `JTable` object called `displayTable` is created with the `detail` and `headersName` arrays as arguments, and a `JScrollPane` object called `scrollPane` is created with `displayTable` as an argument.

The `scrollPane` is added to the `displayPanel`, which is then added to the `JFrame`. When an object of the `DisplayCreditCard` class is instantiated, it displays a window with a table containing the details of all credit cards in the input `ArrayList<BankCard> bankCards`.

## 4.8 Method Description For Display Debit Card

This code defines a class named `DisplayDebitCard` that extends the `JFrame` class. It takes an `ArrayList` of `BankCard` objects as input to its constructor. It is responsible for displaying a table of debit card details in a GUI window.

The constructor sets the title, default close operation, size, location, and visibility of the window. It creates a `JPanel` object to hold the table and sets its layout to a vertical box layout. It creates an array of `String` objects to store the header names for the table. It creates a new `ArrayList` of `DebitCard` objects and adds all `BankCard` objects that are instances of `DebitCard` to it using an enhanced for loop and `instanceof` operator.

It then creates a two-dimensional array of `String` objects to store the details of each `DebitCard` object, and populates the array with the details using another for loop. Finally, it creates a `JTable` object and sets its data and header using the two-dimensional array and header names. It creates a `JScrollPane` object to allow scrolling of the table and adds it to the `JPanel` object. The `JPanel` object is then added to the `JFrame` object.

#### 4.9 Method description for Set Credit Limit

The `SetCreditLimit` class is a Swing GUI-based window that allows a user to set a credit limit for a credit card in a bank's system. The class extends the `JFrame` class and sets up the window with a title, size, position, and visibility. It also contains various Swing components such as panels, labels, text fields, and buttons.

The `SetCreditLimit` constructor takes an `ArrayList` of `BankCard` objects as a parameter. The `BankCard` class is a superclass for `CreditCard` and `DebitCard` classes, which represent the different types of cards a bank can issue to its customers.

The constructor sets up the Swing components required to display the window. It creates a panel to hold all the components and sets its layout to null. It then adds various labels, text fields, and buttons to the panel, each with a specific position and size.

The `cardIdTextFieldSetCreditLimit` is a text field that allows the user to input the ID of the credit card they want to set a limit for. The `creditLimitTextField` is another text field that allows the user to input the desired credit limit for the card. Finally, the `gracePeriodSetCreditLimitTextField` is a text field that allows the user to input the grace period for the credit card.

The `setCreditLimitBtn` is a button that, when clicked, sets the credit limit for the card with the specified ID to the value entered in the `creditLimitTextField` and the grace period to the value entered in the `gracePeriodSetCreditLimitTextField`. The method `CreditCardFound` is called to find the `CreditCard` object with the specified ID in the `bankCards` list. If the card is found, its credit limit is set using the `setCreditLimit` method of the `CreditCard` class.

The `setCreditLimitClearBtn` is a button that clears all the text fields when clicked.

Overall, the `SetCreditLimit` class provides a simple GUI-based interface for users to set credit limits for their credit cards, helping them manage their finances effectively.

#### 4.10 Method description For Main

This is the implementation of the `Main` class, which extends `JFrame` and creates the main window of the application.

The `Main` class has several instance variables that represent the components of the main window, including buttons and a list of bank cards. It also has a constructor that sets up the window and adds the necessary components.

The `mainPanel` is a `JPanel` that represents the main area of the window. It has a titled border with a custom color, and it contains several buttons that allow the user to add, withdraw, and cancel bank cards. Each button has an action listener that opens a new window when clicked.

The `addDebitCardButton` and `addCreditCardButton` buttons open windows that allow the user to add new bank cards to the system. The `withdrawalBtn` button opens a window that allows the user to withdraw money from a debit card. The `setCreditLimitBtn` button opens a window that allows the user to set the credit limit for a credit card. The `cancelCreditCardBtn` button opens a window that allows the user to cancel a credit card. The `DisplayButton` button is not used in the implementation of this class.

Overall, this class creates the main window of the application and sets up the necessary components and listeners to allow the user to interact with the system.

#### 4.11 Method Description For Withdraw From Debit Card

This is a Java class named `WithdrawFromDebitCard` which extends the `JFrame` class. It creates a GUI for withdrawing money from a debit card. The GUI includes labels, text fields, drop-down lists, and buttons for the user to enter the debit card details and perform the withdrawal operation.

The class imports `BankCard` and `DebitCard` classes along with some custom components to be used in the GUI. The constructor of the class takes an `ArrayList` of `BankCard` objects as a parameter.

The `WithdrawFromDebitCard` class creates a new `JPanel` object called `withdrawalPanel` and sets its size, location, border, and layout. The panel contains labels and text fields to accept the debit card details such as card ID, PIN number, withdrawal amount, and date of withdrawal. The date of withdrawal is represented using three drop-down lists for year, month, and day.

Two custom buttons are also created for the Withdrawal and Clear actions.



The GUI is designed using absolute positioning for each component, with the `setBounds()` method to set the location and size of each component. The panel is added to the frame and made visible.

#### 4.12 Method Description for Bank Card

The above code defines a class called "BankCard" that is used to represent a bank card in a banking system. It has several instance variables, including "cardId," "balanceAmount," "bankAccount," "clientName," and "issuerBank," which are used to store information about the card such as its identification number, balance, associated bank account, name of the client, and the issuing bank. The class has a constructor that takes four parameters, "balanceAmount," "cardId," "bankAccount," and "issuerBank," and uses them to initialize the values of the instance variables when an object of the class is created. The class also includes several accessor methods, also known as "getter" methods, which allow other parts of the program to access the values of the instance variables. These are "getCardId," "getBalanceAmount," "getBankAccount," "getClientName," and "getIssuerBank." Additionally, the class has two setter methods, "setClientName" and "setBalanceAmount," that allow the values of the "clientName" and "balanceAmount" instance variables to be updated. The class also includes a method called "display" that prints the values of all instance variables to the console. This method also checks if the "clientName" variable is empty or null and prints a message to the console if it is

#### 4.13 Method Description of Debit Card

This code defines a class called "DebitCard" that extends the "BankCard" class, representing a debit card

in a banking system. It has several instance variables, including "pinNumber," "withdrawlAmount," "dateOfWithdrawl," and "hasWithdrawn," which store information about the debit card such as the PIN number, the amount withdrawn, the date of withdrawal, and whether a withdrawal has been made or not.

The class has a constructor that takes six parameters, "balanceAmount," "cardId," "bankAccount," "issuerBank," "clientName," and "pinNumber," and uses them to initialize the values of the instance variables when an object of the class is created. It also calls the super class constructor (BankCard) and set the clientName using setter method.

The class also includes several accessor methods, also known as "getter" methods, which allow other parts of the program to access the values of the instance variables. These are "getPinNumber," "getWithdrawlAmount," "getDateOfWithdrawl," and "getHasWithdrawn."

Additionally, the class has a setter method "setWithdrawlAmount" that allows the values of the "withdrawlAmount" instance variable to be updated.

The class also includes a method called "WithDraw" that allows the user to withdraw a certain amount from the debit card balance if the entered pin number is correct and the balance is sufficient. It also sets the value of date of withdrawl and hasWithdrawn to true.

The class also includes a "display" method that prints the values of all instance variables to the console and also check if the withdraw has been made or not. It also calls the super class display method.

#### 4.14 Method Description for Credit Card

This code defines a class called "CreditCard" that extends the "BankCard" class, representing a credit card in a banking system. It has several instance variables, including "cvcNumber," "creditLimit,"

"interestRate," "expirationDate," "gracePeriod," and "isGranted," which store information about the credit card such as the CVC number, the credit limit, the interest rate, the expiration date, the grace period, and whether the credit card has been granted or not.

The class has a constructor that takes eight parameters, "cardId," "clientName," "issuerBank," "bankAccount," "balanceAmount," "cvcNumber," "interestRate," and "expirationDate," and uses them to initialize the values of the instance variables when an object of the class is created. It also calls the super class constructor (BankCard) and set the clientName using setter method.

The class also includes several accessor methods, also known as "getter" methods, which allow other parts of the program to access the values of the instance variables. These are "getCvcNumber," "getCreditLimit," "getInterestRate," "getExpirationDate," "getGracePeriod" and "getIsGranted."

Additionally, the class has a setter method "setCreditLimit" that allows the values of the "creditLimit" and "gracePeriod" instance variable to be updated if the credit limit is less than 2.5 times the balance amount.

The class also includes a method called "cancelCreditCard" that allows the user to cancel the credit card and remove the client's credit card if the credit has been granted.

The class also includes a "display" method that prints the values of all instance variables to the console and also check if the credit has been granted or not. It also calls the super class display method.

#### 4.15 Method Description For Bank GUI

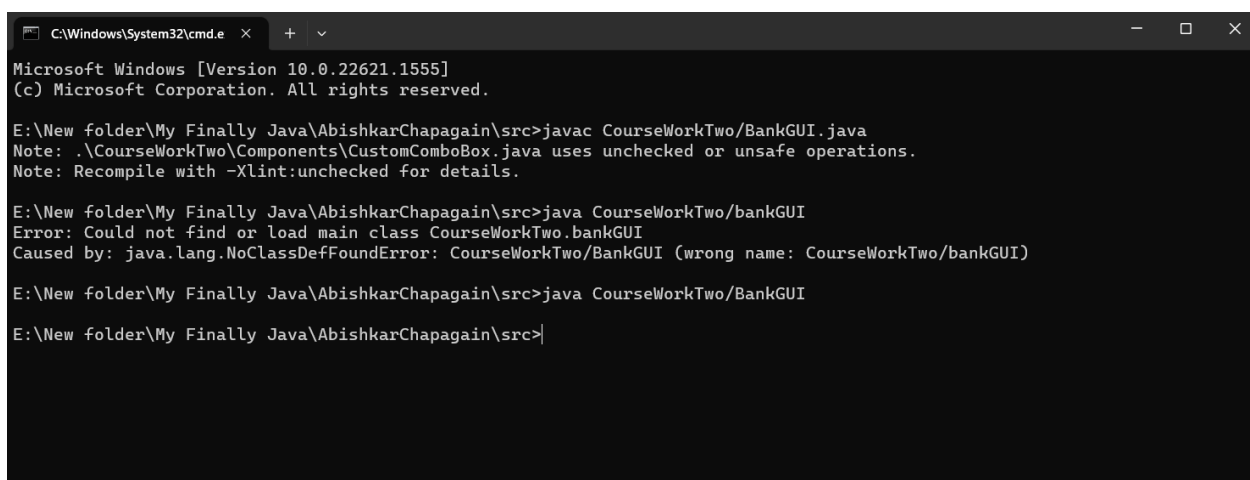
This is a simple Java program that launches a graphical user interface (GUI) for a banking application. The program imports the Main class from the "CourseWorkTwo.views" package and creates an instance of it in the main() method.

The Main class likely contains the code for creating the actual GUI elements, such as buttons, text fields, and labels, and for handling user interactions with those elements. The program may also interact with a backend system to perform banking transactions.

Without more information about the specific implementation of the banking application, it's difficult to say more about how this program works.

## 5. Testing

5.1 Test 1 : Using the command prompt, Testing if the program can be built and executed:



```
C:\Windows\System32\cmd.e  x  +  v
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

E:\New folder\My Finally Java\AbishkarChapagain\src>javac CourseWorkTwo/BankGUI.java
Note: .\CourseWorkTwo\Components\CustomComboBox.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

E:\New folder\My Finally Java\AbishkarChapagain\src>java CourseWorkTwo/bankGUI
Error: Could not find or load main class CourseWorkTwo.bankGUI
Caused by: java.lang.NoClassDefFoundError: CourseWorkTwo/BankGUI (wrong name: CourseWorkTwo/bankGUI)

E:\New folder\My Finally Java\AbishkarChapagain\src>java CourseWorkTwo/BankGUI

E:\New folder\My Finally Java\AbishkarChapagain\src>
```

Figure 6 Compiling and running program through command prompt

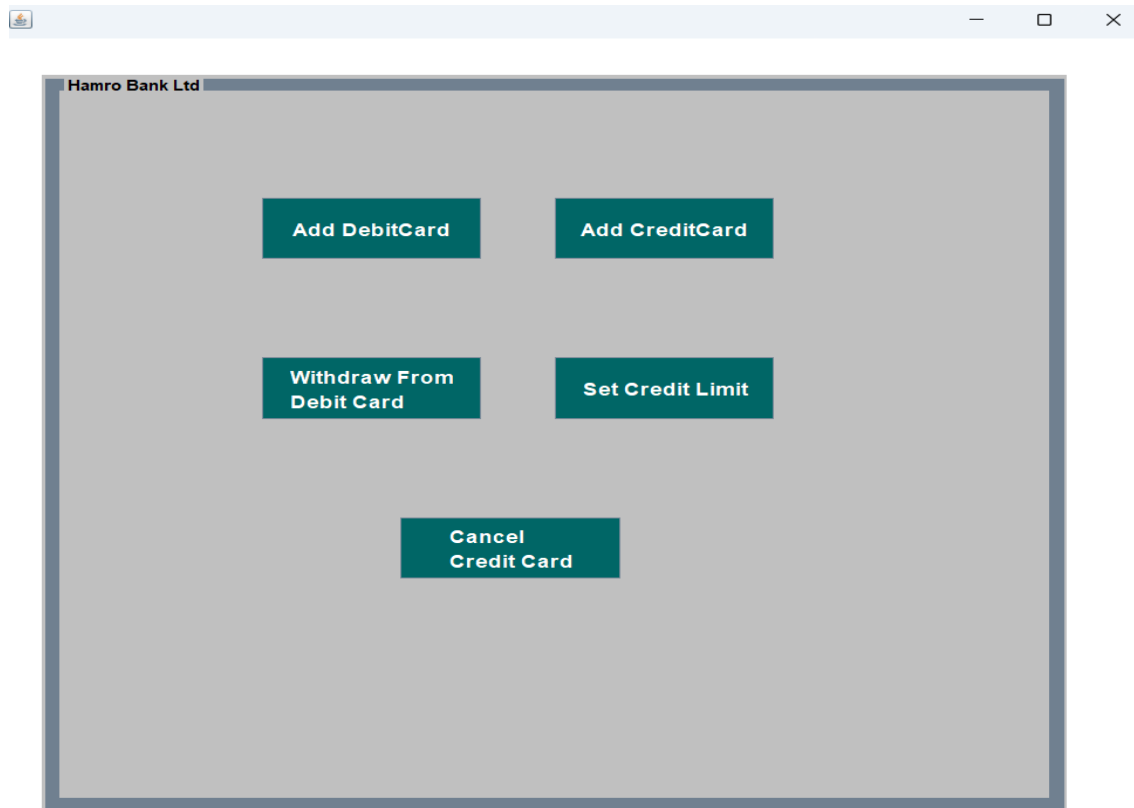


Figure 7 The GUI is displayed after compiling and running the program in the command prompt

## 6 Test Table 1

Objective:	To determine whether a program may be compiled and executed in the command line
Action:	executing the application in the command prompt after compilation.
Expected output:	The GUI Must be displayed
Actual Output	The GUI has displayed
Test Result	Test Sucessful

## 5.2 Evidence of test :

### A. Main Page

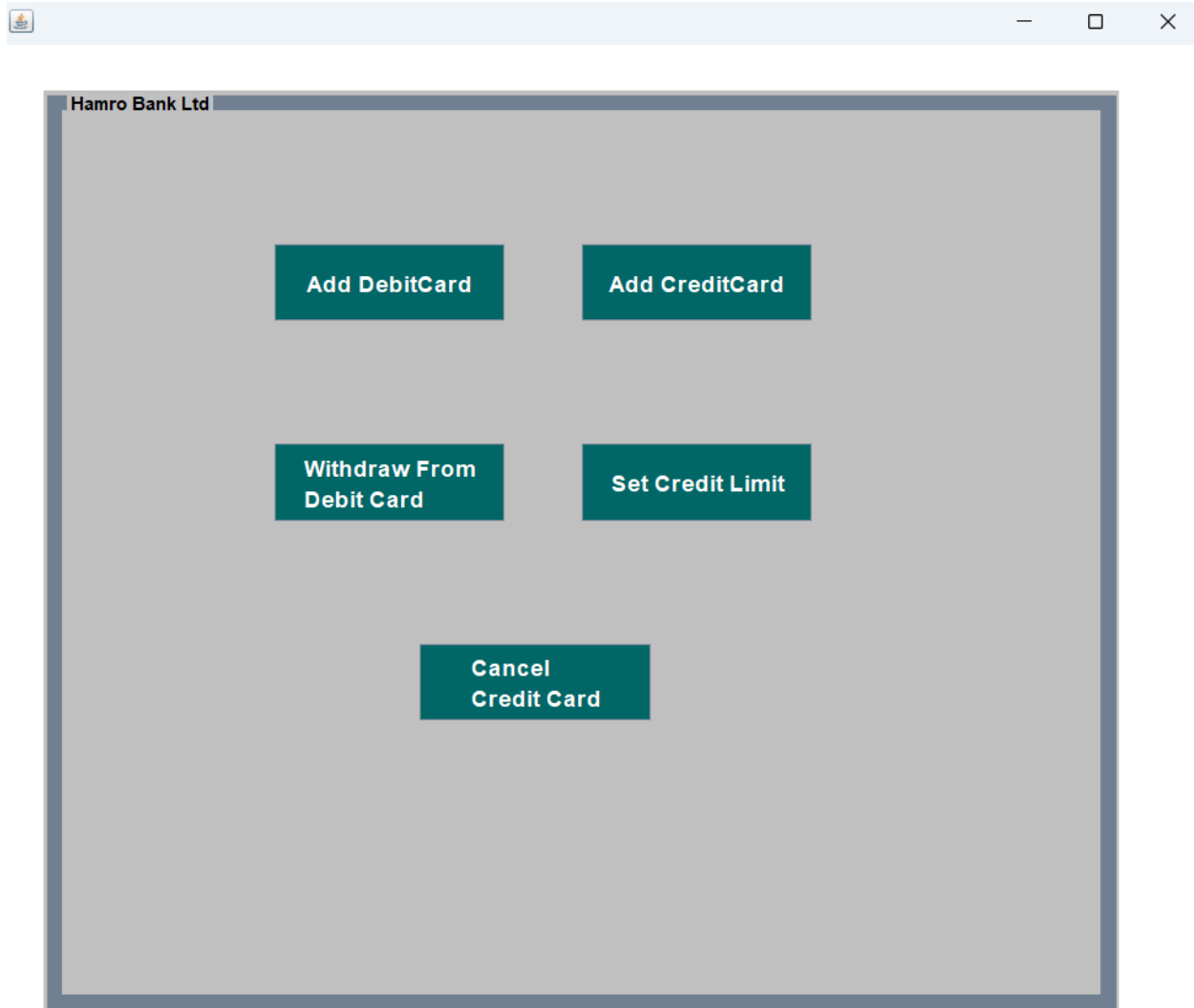


Figure 8 displaying home

### B. Add to debit card

The screenshot shows a Java Swing window titled "Add Debit Card". Inside the window, there is a form with the following fields and labels:

- Card ID**: Text field containing "123"
- PIN Number**: Text field containing "321"
- Client Name**: Text field containing "Abishkar Chapagain"
- Issuer Bank**: Text field containing "mega"
- Bank Account**: Text field containing "1a2s3d"
- Balance Amount**: Text field containing "258631"

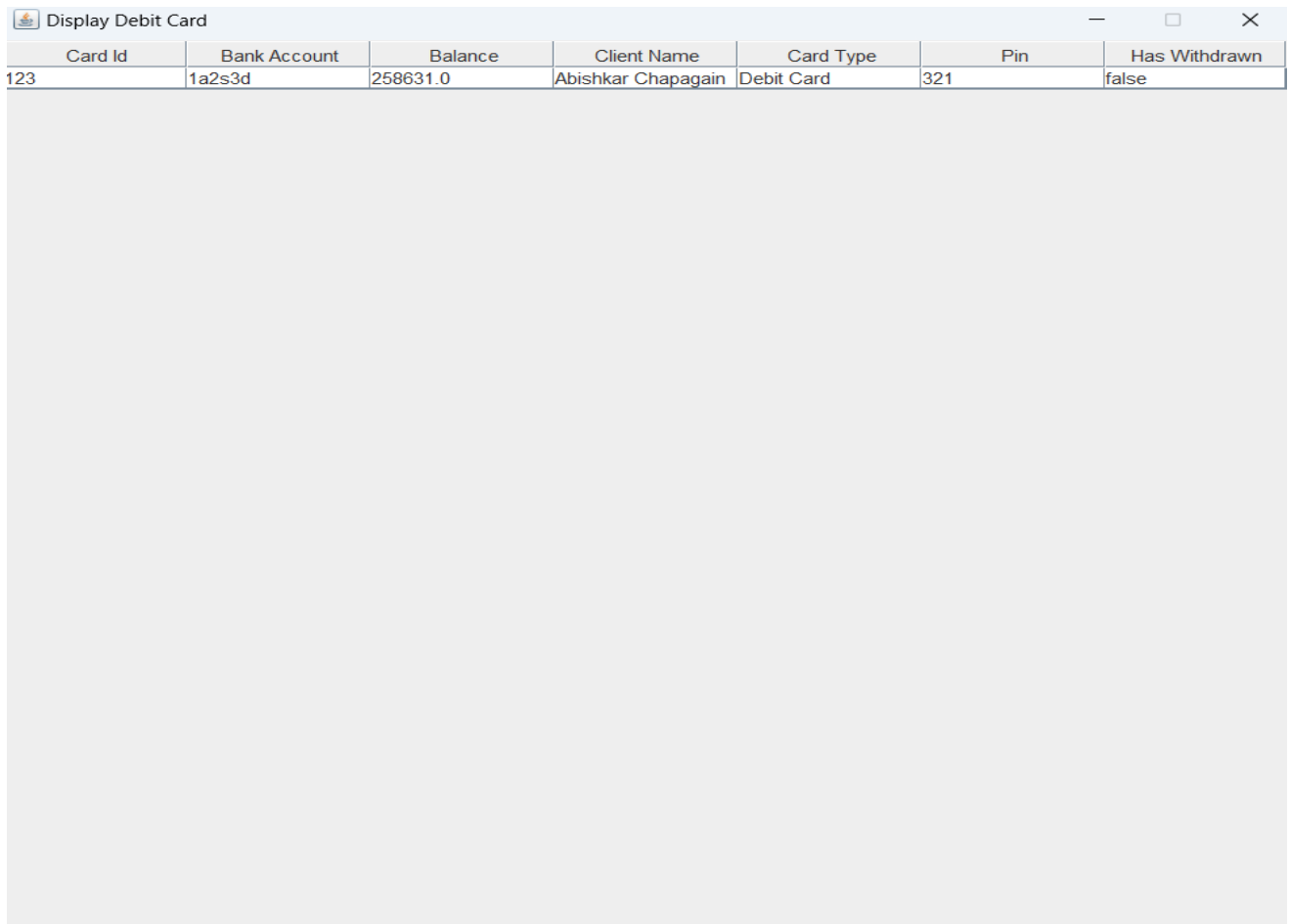
Below the form, there are three buttons:

- Clear**: A green button with white text.
- Display**: A green button with white text.
- Add To Debit Card**: A green button with white text, located at the bottom right of the window.

A message dialog box is displayed in the center of the window. It has a title bar that says "Message" and a close button (X). The dialog box contains an information icon (i) and the text "Debit Card added successfully". Below the text is an "OK" button.

Figure 9 Adding Debit Card

Display After Debit Card Added:



Card Id	Bank Account	Balance	Client Name	Card Type	Pin	Has Withdrawn
123	1a2s3d	258631.0	Abishkar Chapagain	Debit Card	321	false

Figure 10 Display After Adding Debit Card

## 7 Test Table 2:



Objective	<i>To check if Debit Card is added properly or not.</i>
Action:	Clicking the 'Add Card' button after entering required fields.
Expected Output:	The dialog box must display the message,"Debit Card Added Successfully"
Actual Output:	The dialog box displayed the message "Debit Card Added Successfully"
Test Result:	Test Successful.

### C. Add Credit Card

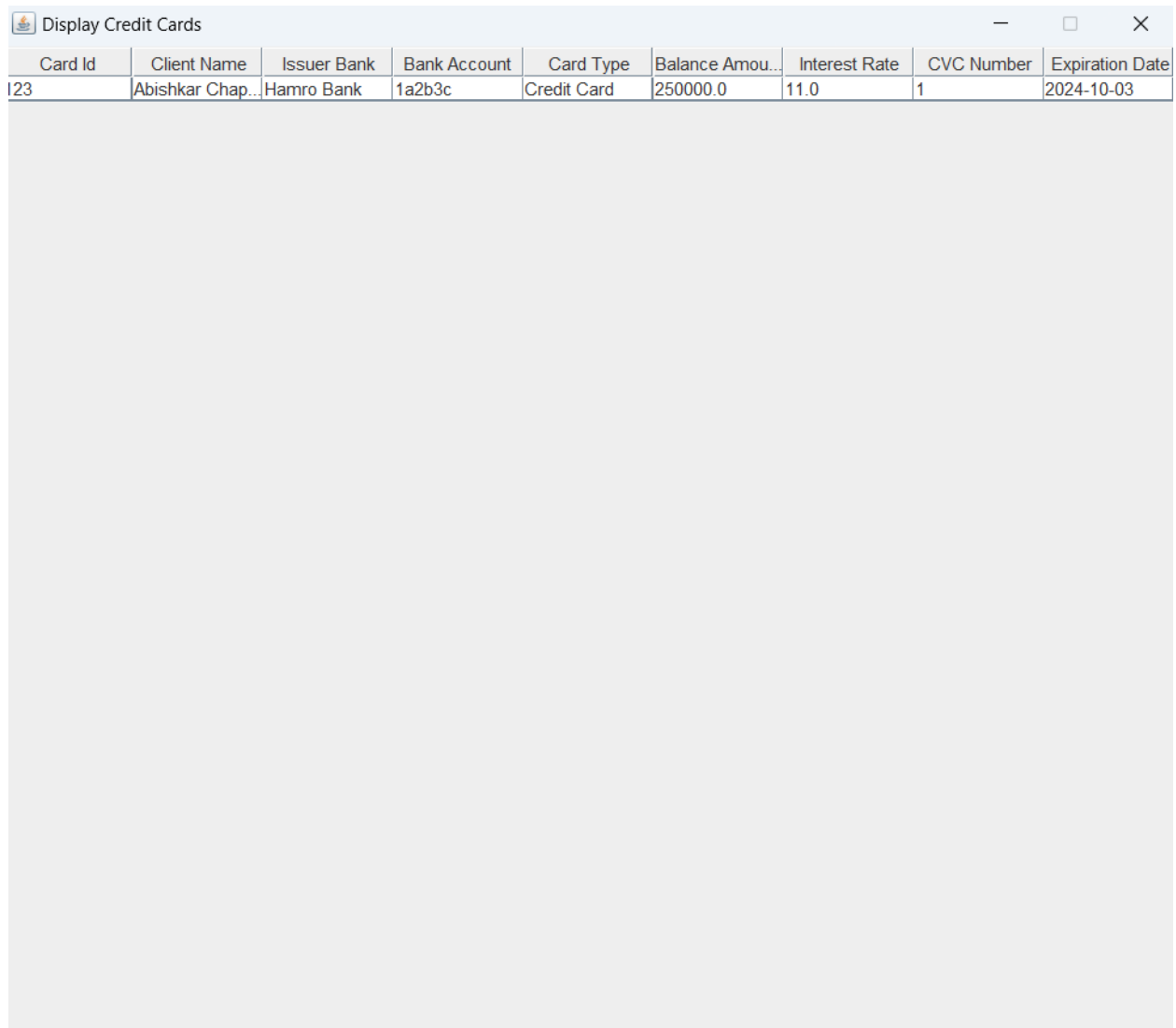
The screenshot shows a Java Swing window titled "Add Credit Card". The window contains a form with the following fields and controls:

- Card ID:** Text input field containing "123".
- CVC Number:** Text input field containing "001".
- Issuer Bank:** Text input field containing "Ham ro Bank".
- Bank Account:** Text input field containing "1a2b3c".
- Balance Amount:** Text input field containing "250000".
- Client Name:** Text input field containing "Abishkar Chapagain".
- Interest Rate:** Text input field containing "11".
- Expiration Date:** Three dropdown menus showing "2024", "10", and "03".
- Buttons:** "Clear", "Display", and "Add To Credit Card" (located at the bottom right).

A message dialog box is displayed in the center of the window, titled "Message". It contains an information icon and the text "Adding a credit card was successful." with an "OK" button.

Figure 11 Adding credit Card

Display the details of Credit Card



The screenshot shows a window titled "Display Credit Cards" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window is a table with 9 columns. The first row of the table contains the following data: Card Id: 123, Client Name: Abishkar Chap..., Issuer Bank: Hamro Bank, Bank Account: 1a2b3c, Card Type: Credit Card, Balance Amou...: 250000.0, Interest Rate: 11.0, CVC Number: 1, and Expiration Date: 2024-10-03. The rest of the table area is empty.

Card Id	Client Name	Issuer Bank	Bank Account	Card Type	Balance Amou...	Interest Rate	CVC Number	Expiration Date
123	Abishkar Chap...	Hamro Bank	1a2b3c	Credit Card	250000.0	11.0	1	2024-10-03

Figure 12 Display after adding Credit Card

## 8 Test Table 3

Objective	To Check If Credit Card Is added properly or not
Action	Clicking the ' ADD CARD button after entering required fields.'
Expected Output	The dialog box must display the message,"Credit Card Added Successfully"
Actual Output	The dialog box displayed the message "Credit Card Added Successfully"
Test Result	Test Successful.

### D. Withdraw From Debit Card

The screenshot shows a web application window titled "Withdraw from Debit Card". Inside the window, there are four input fields: "Card ID:" with the value "111", "PIN Number:" with the value "123", "Withdrawal Amount:" with the value "258963", and "Date Of Withdrawal:" with a date picker set to "2019-08-09". Below these fields are two buttons: "Clear" and "Display". A "Withdrawal" button is located at the bottom right. A modal message box is displayed in the center, titled "Message", with an information icon and the text "Your account has been successfully debited with the withdrawn amount :)", and an "OK" button.

Figure 13 Entering correct Pin number and Card Id to withdraw

## Display After Withdraw

Display Debit Card						
Card Id	Bank Account	Balance	Client Name	Card Type	Pin	Has Withdrawn
123	1a2s3d	258631.0	Abishkar Chapagain	Debit Card	321	false
111	ghj	2.58741110889E11	daf	Debit Card	123	true

Figure 14 Display after withdraw

## 9 Test Table 4

<b>Objective:</b>	To check after withdraw amount will deducted or not
<b>Action:</b>	Clicking the 'withdraw' after entering correct pin number and Card Id.
<b>Expected Output:</b>	The dialog box must display the message,"Your account has been successfully debited with the withdrawn amount"
<b>Actual Output:</b>	The dialog box displayed the message " Your account has been successfully debited with the withdrawn amount "
<b>Test Result:</b>	Test Successful.

E. Set credit limit

Withdraw from Debit Card

**Set Credit Limit**

Card ID

123

Credit Limit

256

Grace Period

10

Clear

Message

The system has updated your credit limit successfully:)

OK

Set Limit

Figure 15set credit limit

## F. Cancel Credit card

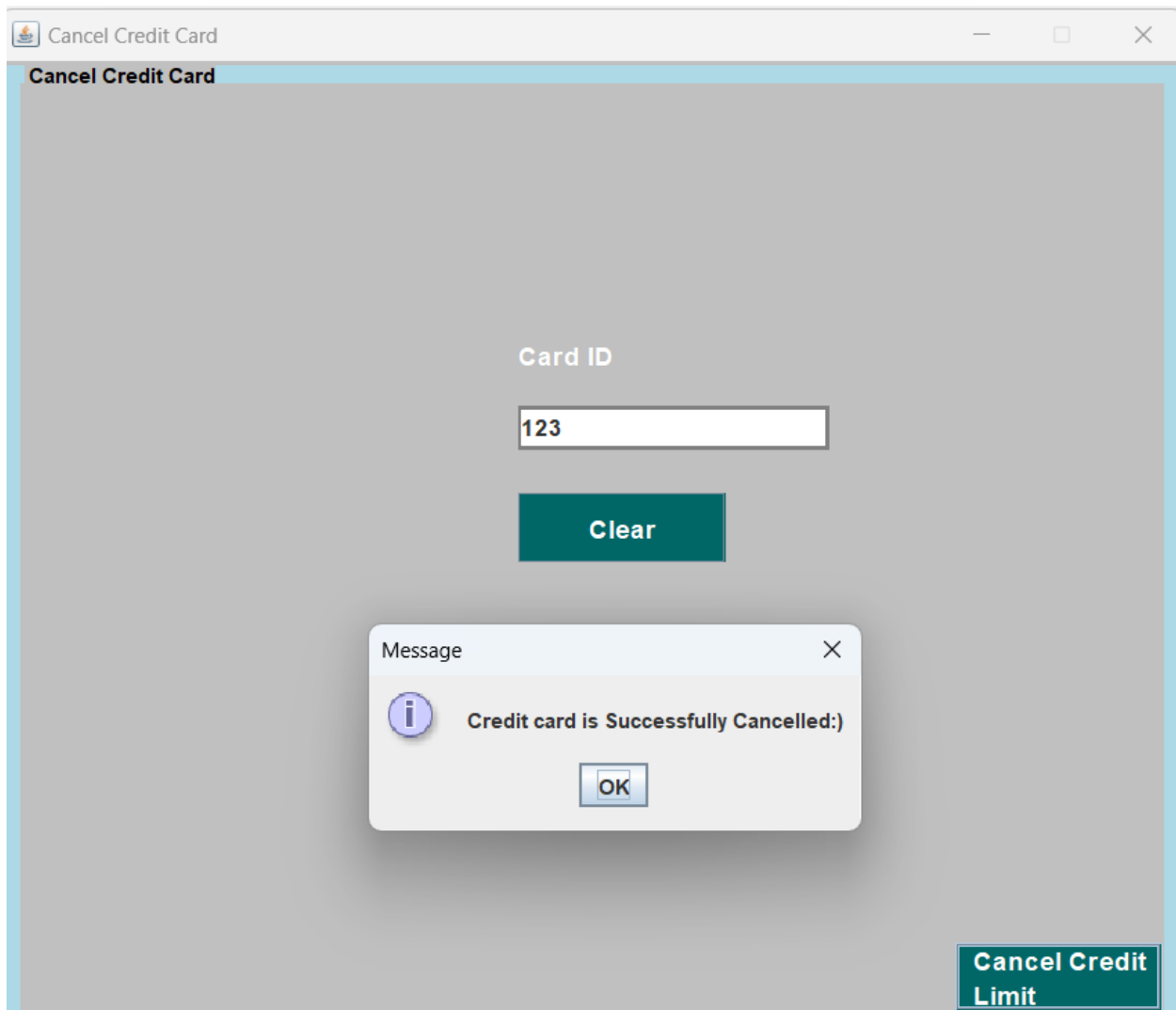
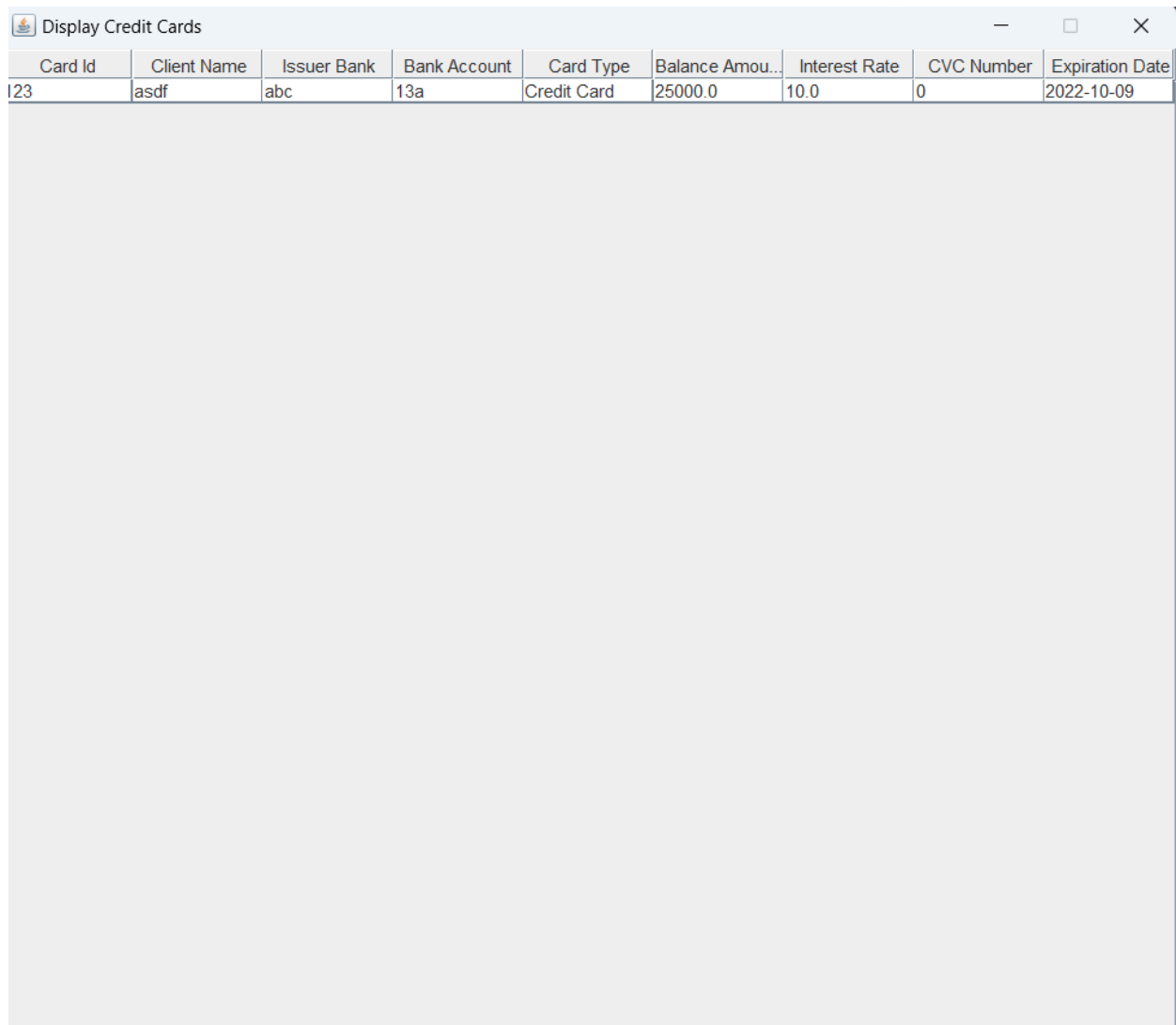


Figure 16canceling the credt limit

After cancelling drecit limit cvc number is being 0





The screenshot shows a window titled "Display Credit Cards" with a table containing one row of data. The table has the following columns: Card Id, Client Name, Issuer Bank, Bank Account, Card Type, Balance Amou..., Interest Rate, CVC Number, and Expiration Date. The data row contains the values: 123, asdf, abc, 13a, Credit Card, 25000.0, 10.0, 0, and 2022-10-09.

Card Id	Client Name	Issuer Bank	Bank Account	Card Type	Balance Amou...	Interest Rate	CVC Number	Expiration Date
123	asdf	abc	13a	Credit Card	25000.0	10.0	0	2022-10-09

Figure 17 Details after canceling credit limit

## 10 Test Table 4

*Figure 17 Display after Setting Credit Limit*

<b>Objective:</b>	To verify whether your credit limit has been set successfully!
<b>Action:</b>	Clicking the 'set limit ' button after entering correct Card Id.
<b>Expected Output:</b>	The dialog box must display the message,"The System have updated your credit limit successfully"
<b>Actual Output:</b>	The dialog box displayed the message "The System have updated your credit limit successfully"
<b>Test Result:</b>	Test Successful.

### 6. Test 3

## 11

## 6.1 Putting value error

The screenshot shows a Java Swing window titled "Add Debit Card". Inside the window, there is a form with six text input fields arranged in two columns. The left column contains "Card ID" (with value "-100"), "Client Name" (with value "abishkar"), and "Bank Account" (with value "256482"). The right column contains "PIN Number" (with value "100"), "Issuer Bank" (with value "abc"), and "Balance Amount" (with value "25146"). Below the "Card ID" field, there are two buttons: "Clear" and "Display". At the bottom right of the window is a button labeled "Add To Debit Card". A modal message dialog box is centered over the form. It has a title bar "Message" and a close button (X). The dialog contains an information icon (i) and the text "Invalid card ID. Please enter a positive integer." Below the text is an "OK" button.

Figure 18keeping int value less than 0 and mwssage was shown

## 6.2 Credit card

The screenshot shows a Java Swing window titled "Add Credit Card". Inside the window, there are two columns of input fields. The left column contains "Card ID:" with a text box containing "-111", "Issuer Bank" with a text box containing "hjpg", "Balance Amount:" with a text box containing "25142", and "Interest Rate" with a text box containing "20". The right column contains "CVC Number:" with a text box containing "123", "Bank Account:" with a text box containing "145", "Client Name" with a text box containing "abish", and "Expiration Date" with three dropdown menus showing "2016", "07", and "08". At the bottom left are two green buttons: "Clear" and "Display". At the bottom right is a green button: "Add To Credit Card". A modal message box is displayed in the center, titled "Message", with an information icon and the text "Card ID is invalid. Enter a positive number only." and an "OK" button.

Field	Value
Card ID	-111
CVC Number	123
Issuer Bank	hjpg
Bank Account	145
Balance Amount	25142
Client Name	abish
Interest Rate	20
Expiration Date	2016-07-08

Buttons: Clear, Display, Add To Credit Card

Message: Card ID is invalid. Enter a positive number only.

Figure 19value error on credit card and message shown

## 6.3 WITHDRAW

The screenshot shows a web application window titled "Withdraw from Debit Card". Inside the window, there is a form with the following fields and controls:

- Card ID:** A text input field containing the value "123".
- PIN Number:** A text input field containing the value "123".
- Withdrawal Amount:** A text input field containing the value "1548458200".
- Date Of Withdrawal:** Three dropdown menus showing the date "2020", "08", and "06".
- Buttons:** There are two buttons on the left labeled "Clear" and "Dis" (partially visible), and a "Withdrawal" button at the bottom right.

An error message dialog box is displayed in the foreground, titled "Message". It contains an information icon and the text: "The account does not have sufficient funds to process the requested transaction!!". There is an "OK" button at the bottom of the dialog.

Figure 20 Trying to withdraw more than money then entered in debit card

12

## 6.4 Set Credit Limit

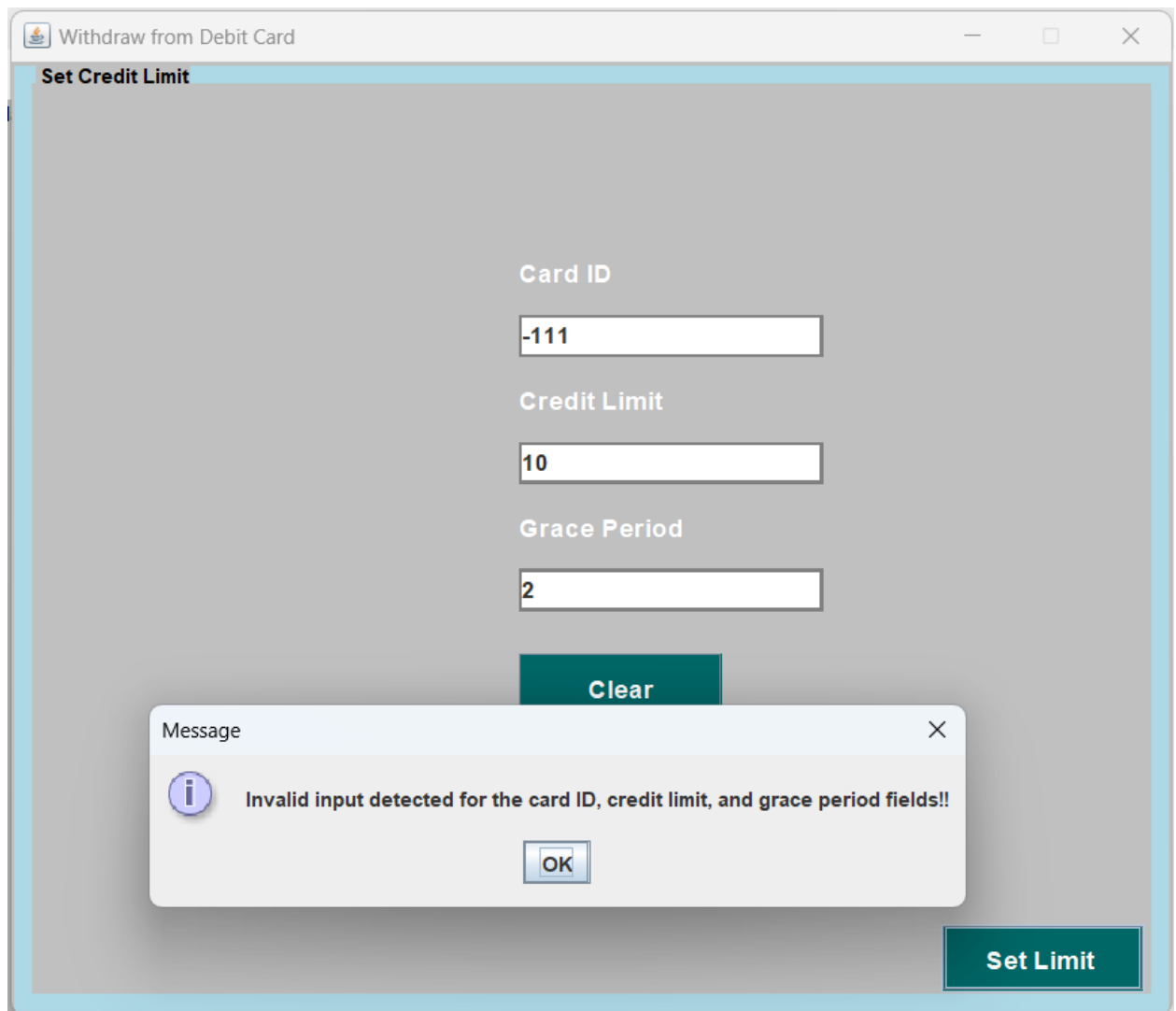


Figure 21 Entering negative in card id and message shown

13

13.5 cancel credit limit

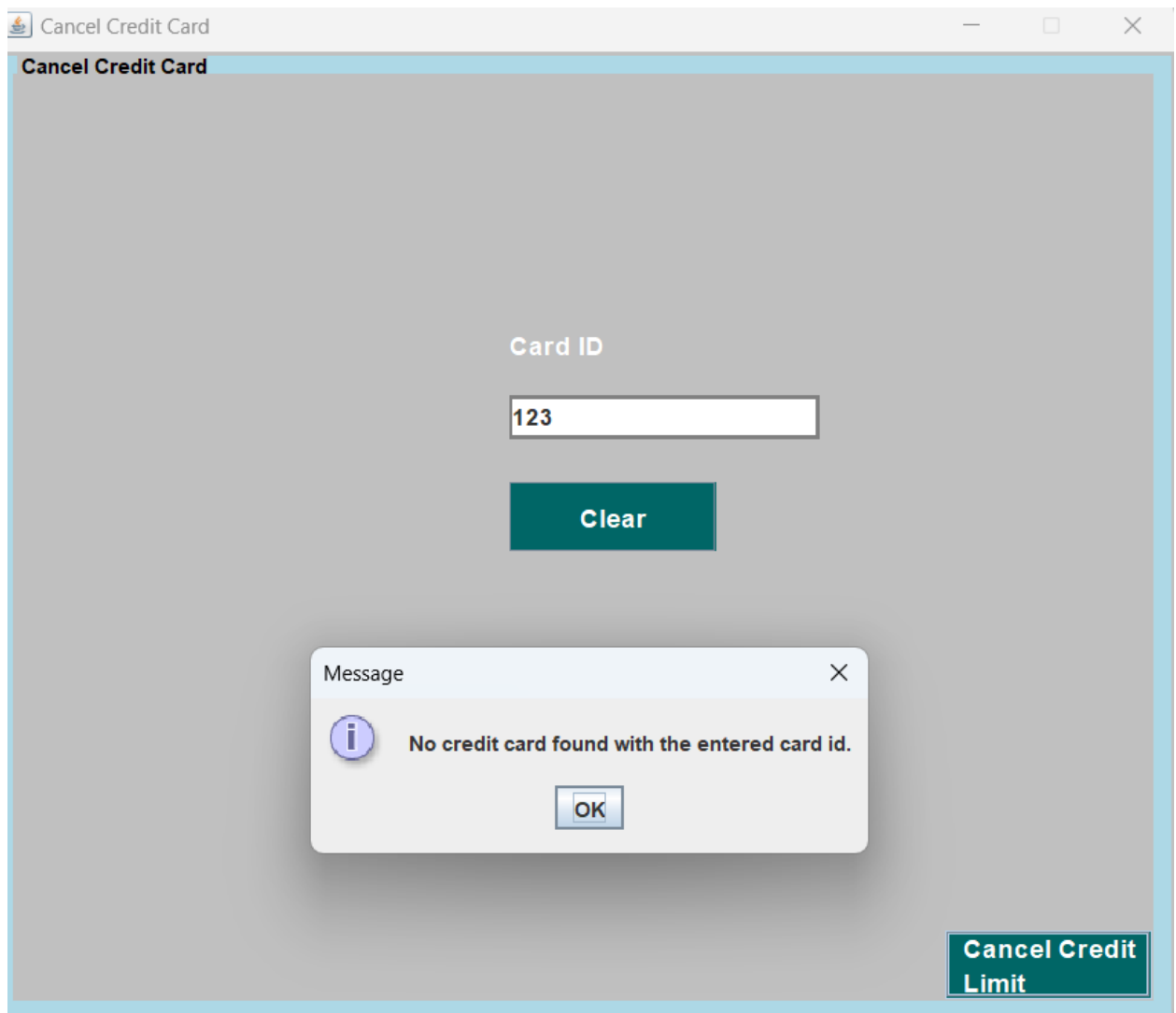


Figure 22 Entering wrong Card Id in Cancel credit limit

14

## 15 Error Detection and Correction

I ran into a number of mistakes while developing my application, but I was able to spot and correct them. I've included a few instances of the mistakes I found and updated underneath.

## 6.1 Syntax Error:

Syntax errors can occur when punctuations such as commas, curly braces, semi-colons, etc. are omitted or when variable names are misspelled.

### 6.1.1 Error detection:

As shown in the figure below, the error is caused by the missing semi-colon (;) in the highlighted statement, which prevents the code from being compiled.

A screenshot of a code editor showing Java code. The code is for a class named 'WithdrawFromDebitCard' that inherits from 'JFrame'. The code includes several method calls: 'super()', 'setSize()', 'setDefaultCloseOperation()', 'setLocationRelativeTo()', 'setResizable()', and 'setVisible()'. The line 'setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE)' is highlighted, and a red squiggly line indicates a syntax error at the end of the line, specifically the missing semi-colon. Below this, there is a comment '//Creating panel for withdrawal from debit card' followed by the creation and configuration of a 'JPanel' object named 'withdrawalPanel'.

```
1 usage
public WithdrawFromDebitCard(ArrayList<BankCard> bankCards) {
    super( title: "Withdraw from Debit Card");
    setSize( width: 700, height: 600);
    setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    setLocationRelativeTo(null);
    setResizable(false);
    setVisible(true);

    //Creating panel for withdrawal from debit card
    JPanel withdrawalPanel = new JPanel();
    withdrawalPanel.setBounds(x: 30, y: 30, width: 630, height: 500);
    withdrawalPanel.setBorder(new TitledBorder(new LineBorder(new Color( r: 173, g: 216, b: 230), thickness: 10), title: "Withdrawal From Debi
    //(.setBorder) creating TitledBorder,LineBorder and Color object to add colorFull title with line border at Top and center
```

Figure 23 Syntax Error occurred due to missing of semi-colon

### 6.1.2 Error correction:

In the figure below, error found due to missing of semi-colon is corrected by adding semi-colon after the statement as shown in the highlighted line.



```

1 usage
public WithdrawFromDebitCard(ArrayList<BankCard> bankCards) {
    super( title: "Withdraw from Debit Card");
    setSize( width: 700, height: 600);
    setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    setLocationRelativeTo(null);
    setResizable(false);
    setVisible(true);

    //Creating panel for withdrawal from debit card
    JPanel withdrawalPanel = new JPanel();
    withdrawalPanel.setBounds( x: 30, y: 30, width: 630, height: 500);
    withdrawalPanel.setBorder(new TitledBorder(new LineBorder(new Color( r: 173, g: 216, b: 230), thickness: 10), title: "Withdrawal From Debi
    //(.setBorder) creating TitledBorder,LineBorder and Color object to add colorFull title with line border at Top and center

```

Figure 24 Syntax error correction by adding semi-colon

## 6.2 Runtime Error

The error that takes place while executing a program is runtime error. This kind of error is encountered while running the program. This kinds of errors are intermediately difficult to find.

### 6.2.1 Error Detection:

```
withdrawalPanel.setBounds(x: 30, y: 30, width: 630, height: 500);
withdrawalPanel.setBorder(new TitledBorder(new LineBorder(new Color(r: 173, g: 216, b: 173))));
//(.setBorder) creating TitledBorder,LineBorder and Color object to add colorFull titl
withdrawalPanel.setBackground(lightGray);
withdrawalPanel.setLayout();
add(withdrawalPanel);

//creating label and textField for CardId Withdrawal
cardIdLabelWithdrawal = new MyCustomLabel(text: "Card ID:");
cardIdLabelWithdrawal.setBounds(x: 50, y: 115, width: 140, height: 20);
withdrawalPanel.add(cardIdLabelWithdrawal);
```

Figure 25 something is missing in set layout

```
E:\New folder\My Finally Java\AbishkarChapagain\src\CourseWorkTwo\views\WithdrawFromDebitCard.java:43:24
java: method setLayout in class java.awt.Container cannot be applied to given types;
    required: java.awt.LayoutManager
    found:    no arguments
    reason: actual and formal argument lists differ in length
```

Figure 26 pRblems on RunTime Error

### 6.2.2 Error Correction:

```

withdrawalPanel.setBackground(lightGray);
withdrawalPanel.setLayout(null);
add(withdrawalPanel);

//creating label and textField for CardId Withdrawal
cardIdLabelWithdrawal = new MyCustomLabel( text: "Card ID:");
cardIdLabelWithdrawal.setBounds( x: 50, y: 115, width: 140, height: 20);
withdrawalPanel.add(cardIdLabelWithdrawal);

```

Figure 27 Null was missing in SetLayout

### 6.3 Logical Error

When the program compiles and runs but doesn't give the desired result. Then, such types of error is logical error.

```

    if (!cardFound) {
        JOptionPane.showMessageDialog( parentComponent: null, message: "The card you entered cannot be found in the system.");
    } else if (withdrawal > 0) {
        JOptionPane.showMessageDialog( parentComponent: null, message: "Invalid withdrawal amount. Please enter a positive number");
    } else {
        Withdraw(debitCard, withdrawalPin, withdrawal, dateOfWithdrawal);
    }
} catch (Exception err) {
    JOptionPane.showMessageDialog( parentComponent: null, message: "Invalid input format. Please enter a valid number for card ID");
}

```

Figure 28 errann (>) sign

#### 6.3.1 Error Detection:

The screenshot shows a web application window titled "Withdraw from Debit Card". The main form has the following fields and controls:

- Card ID:** A text input field containing "123".
- PIN Number:** A text input field containing "123".
- Withdrawal Amount:** A text input field containing "-111".
- Date Of Withdrawal:** Three dropdown menus showing "2018", "04", and "03".
- Buttons:** "Clear", "Display", and "Withdrawal" (located at the bottom right).

A modal message box is displayed in the center, titled "Message". It contains an information icon and the text: "Your account has been successfully debited with the withdrawn amount :)". An "OK" button is at the bottom of the message box.

Figure 29 due to > sign withdrawn amount is >0

### 6.3.2 Error Correction

```

    }
    if (!cardFound) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "The card you entered cannot be found in the system.");
    } else if (withdrawal ≤ 0) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Invalid withdrawal amount. Please enter a positive number.");
    } else {
        Withdraw(debitCard, withdrawalPin, withdrawal, dateOfWithdrawal);
    }
} catch (Exception err) {
    JOptionPane.showMessageDialog(parentComponent: null, message: "Invalid input format. Please enter a valid number for card ID, wit

```

Figure 30 Logical error was missing and which was corrected

## 7. Conclusion

The second programming coursework for our first-year module was assigned in the 20th week, and it required us to add a new class to the project we created in the first part of the coursework. The new class was used to create a graphical user interface for a banking management system. Initially, I found GUI designing to be easy, but I was mistaken. I encountered various problems, including difficulty with layout managers and component positioning, and incorrect button functions. Despite the challenges, with the help of my module leader and friends, I was able to complete the coursework.

Working on the banking management system's graphical user interface allowed me to learn many new things beyond what was covered in my regular classes. I conducted extensive research on GUI and its contents, which allowed me to practice and implement my understanding of Java GUI. Additionally, I gained a deeper understanding of layout managers, exception handling, and class casting. Throughout this assessment, I also developed my research and problem-solving skills. While working on the interface, I faced errors and frustration, but this motivated me to do more research and revise my lecture materials on Java GUI. My subject teacher was also very helpful in resolving the errors I encountered.

Completing the second programming coursework helped me develop several skills. I learned how to design a graphical user interface using Java GUI and gained a deeper understanding of layout managers, exception handling, and class casting. Additionally, I developed my research and problem-solving skills while facing errors and frustration during the coursework. I also appreciated the support I received from my module leader and friends, which helped me complete the coursework successfully. Overall, the second programming coursework was a challenging but rewarding experience that allowed me to gain new skills and deepen my understanding of Java GUI.

## 8. Appendix

### 8.1. CustomComboBox

```
package CourseWorkTwo.Components;

import javax.swing.*;
import java.awt.*;

public class CustomComboBox extends JComboBox {

    public CustomComboBox(String[] values) {
        super(values);
        setFont(new Font("Tohama", Font.BOLD, 15));
        setMaximumRowCount(5);
    }
}
```

### 8.2 CustomButton

```
package CourseWorkTwo.Components;

import javax.swing.*;
import java.awt.*;

public class MyCustomBtn extends JButton {

    public MyCustomBtn(String text) {
        super(text);
        setFocusPainted(false);
        setContentAreaFilled(true);
        setFont(new Font("Dialog", Font.BOLD, 15));
        setForeground(new Color(255, 255, 255));
        setBackground(new Color(0, 102, 102));
        setMargin(new Insets(10, 10, 10, 10));
    }
}
```

```
}  
}
```

### 8.3 CustomLabel

```
package CourseWorkTwo.Components;  
  
import javax.swing.*;  
import java.awt.*;  
  
public class MyCustomLabel extends JLabel {  
    public MyCustomLabel(String text){  
        super(text);  
        setFont(new Font("Tohama", Font.BOLD, 15));  
        setForeground(Color.white);  
    }  
}
```

### 8.4. MyCustomTextField

```
package CourseWorkTwo.Components;  
  
import javax.swing.*;  
import java.awt.*;  
  
public class MyCustomTextField extends JTextField {  
  
    public MyCustomTextField(String text){  
        super(text);  
        setColumns(10);  
        setBorder(BorderFactory.createLineBorder(Color.gray, 2));  
        setFont(new Font("Arial", Font.BOLD, 14));  
    }  
}
```

### 8.5 Add Credit Card

```
package CourseWorkTwo.views;  
  
import CourseWorkTwo.BankCard;  
import CourseWorkTwo.Components.CustomComboBox;
```

```

import CourseWorkTwo.Components. MyCustomBtn;
import CourseWorkTwo.Components.MyCustomLabel;
import CourseWorkTwo.Components.MyCustomTextField;
import CourseWorkTwo.CreditCard;
import CourseWorkTwo.DebitCard;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.util.ArrayList;

import static java.awt.Color.*;

public class AddCreditCard extends JFrame {

    MyCustomLabel cardIdLabel, clientNameLabel, issuerBankLabel, bankAccountLabel,
    balanceAmountLabel, CVCNumberLabel, interestRateLabel, expirationDateLabel;
    MyCustomTextField cardIdTextField, clientNameTextField, issuerBankTextField,
    bankAccountTextField, balanceAmountTextField, CVCNumberTextField,
    interestRateTextField;
    CustomComboBox
    expirationDateComboBoxYear, expirationDateComboBoxMonth, expirationDateComboBoxDay;
    MyCustomBtn addCreditLimitBtn, clearButton, displayCreditCardBtn;

    public AddCreditCard(ArrayList<BankCard> bankCards) {
        super("Add Credit Card");
        setSize(700, 600);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(true);
        setVisible(true);

        //Creating panel for Credit card
        JPanel addCreditCardPanel = new JPanel();
        addCreditCardPanel.setBounds(30, 30, 630, 500);
        addCreditCardPanel.setBorder(new TitledBorder(new LineBorder(new Color(173,
216, 230), 10), "Add Credit Card", TitledBorder.LEFT, TitledBorder.TOP, null, new
Color(0,0,0)));
        //(.setBorder) creating TitledBorder,LineBorder and Color object to add
colorFull title with line border at Top and center
        addCreditCardPanel.setBackground(lightGray);
        addCreditCardPanel.setLayout(null);
        add(addCreditCardPanel);

        //creating label and textField for CardID Credit Card
        cardIdLabel =new MyCustomLabel("Card ID:");
        cardIdLabel.setBounds(50,70,140,20);
        addCreditCardPanel.add(cardIdLabel);

        cardIdTextField = new MyCustomTextField("");
        cardIdTextField.setBounds(50,110,180,25);
        addCreditCardPanel.add(cardIdTextField);

        //creating label and textField for CVC Number Credit Card
        CVCNumberLabel =new MyCustomLabel("CVC Number:");
        CVCNumberLabel.setBounds(350,70,140,20);
        addCreditCardPanel.add(CVCNumberLabel);

        CVCNumberTextField = new MyCustomTextField("");
        addCreditCardPanel.add(CVCNumberTextField);
        CVCNumberTextField.setBounds(350,110,180,25);
    }
}

```



```

//creating label and textField for issuerBank Credit Card
issuerBankLabel =new MyCustomLabel("Issuer Bank");
issuerBankLabel.setBounds(50,150,140,20);
addCreditCardPanel.add(issuerBankLabel);

issuerBankTextField = new MyCustomTextField("");
addCreditCardPanel.add(issuerBankTextField);
issuerBankTextField.setBounds(50,190,180,25);

//creating label and textField for BankAccount Credit Card
bankAccountLabel =new MyCustomLabel("Bank Account:");
bankAccountLabel.setBounds(350,150,140,20);
addCreditCardPanel.add(bankAccountLabel);

bankAccountTextField = new MyCustomTextField("");
addCreditCardPanel.add(bankAccountTextField);
bankAccountTextField.setBounds(350,190,180,25);

//creating label and textField for BalanceAmount CreditCard
balanceAmountLabel =new MyCustomLabel("Balance Amount:");
balanceAmountLabel.setBounds(50,230,140,20);
addCreditCardPanel.add(balanceAmountLabel);

balanceAmountTextField = new MyCustomTextField("");
addCreditCardPanel.add(balanceAmountTextField);
balanceAmountTextField.setBounds(50,270,180,25);

//creating label and textField for clientName Credit Card
clientNameLabel =new MyCustomLabel("Client Name");
clientNameLabel.setBounds(350,230,140,20);
addCreditCardPanel.add(clientNameLabel);

clientNameTextField = new MyCustomTextField("");
addCreditCardPanel.add(clientNameTextField);
clientNameTextField.setBounds(350,270,180,25);

//creating label and textField for Interest Rate Credit Card
interestRateLabel =new MyCustomLabel("Interest Rate");
interestRateLabel.setBounds(50,310,140,20);
addCreditCardPanel.add(interestRateLabel);
interestRateLabel.setForeground(white);
interestRateLabel.setFont(new Font("Tohama",Font.BOLD,15));

interestRateTextField = new MyCustomTextField("");
addCreditCardPanel.add(interestRateTextField);
interestRateTextField.setBounds(50,350,180,25);

//creating label and JComboBox(dropdown) for Expiration Date
expirationDateLabel = new MyCustomLabel("Expiration Date");
expirationDateLabel.setBounds(350,310,140,20);
addCreditCardPanel.add(expirationDateLabel);

//ComboBox for years
expirationDateComboBoxYear = new CustomComboBox(new String[]{
    "Year", "2016", "2017", "2018", "2019",
    "2020", "2021", "2022", "2023", "2024",
    "2025", "2026", "2027", "2028", "2029", "2030", "2031", "2032", "2033", "2034", "2 "
});

```

```

expirationDateComboBoxYear.setBounds(350,350,62,25);
addCreditCardPanel.add(expirationDateComboBoxYear);

//ComboBox for months
expirationDateComboBoxMonth = new CustomComboBox(new String[]{
    "Month","01", "02", "03", "04", "05", "06", "07", "08", "09",
    "10", "11", "12"
});
expirationDateComboBoxMonth.setBounds(415,350,72,25);
addCreditCardPanel.add(expirationDateComboBoxMonth);

//ComboBox for days
expirationDateComboBoxDay = new CustomComboBox(new String[]{"Day",
    "01", "02", "03", "04", "05", "06", "07", "08", "09",
    "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
    "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"
});
expirationDateComboBoxDay.setBounds(490,350,55,25);
addCreditCardPanel.add(expirationDateComboBoxDay);

//Creating Add Credit limit button and clear button Credit Card
addCreditLimitBtn = new MyCustomBtn("<html><With>Add To <br>Credit
Card</body></html>");
addCreditCardPanel.add(addCreditLimitBtn);
addCreditLimitBtn.setBounds(553,510,120,40);

//Clear Button for credit card
clearButton = new MyCustomBtn("Clear");
addCreditCardPanel.add(clearButton );
clearButton .setBounds(50,410,120,40);

//Display Button for Credit Card
displayCreditCardBtn =new MyCustomBtn("Display");
addCreditCardPanel.add(displayCreditCardBtn);
displayCreditCardBtn.setBounds(50,470,120,40);

clearButton.addActionListener(e ->{
    clearEntry();
});

addCreditLimitBtn.addActionListener(e ->{
    String cardIdOfCreditCard = cardIdTextField.getText();
    String clientNameOfCreditCard = clientNameTextField.getText();
    String issuerBankCreditCard = issuerBankTextField.getText();
    String bankAccountCreditCard = bankAccountTextField.getText();
    String balanceAmountCreditCard = balanceAmountTextField.getText();
    String interestRateCreditCard = interestRateTextField.getText();
    String CVCNumberCreditCard =CVCNumberTextField.getText();

    if(issuerBankCreditCard.equals("") || clientNameOfCreditCard.equals("") ||
    bankAccountCreditCard.equals("") || cardIdOfCreditCard.equals("") ||
    balanceAmountCreditCard.equals("") || CVCNumberCreditCard.equals("") ||
    interestRateCreditCard.equals("")){
        JOptionPane.showMessageDialog(null,"Before moving further, please make
        sure that all the mandatory fields are filled out.");
    }
}

```

```

        } else if (expirationDateComboBoxYear.getSelectedIndex() == 0 ||
expirationDateComboBoxMonth.getSelectedIndex() == 0 ||
expirationDateComboBoxDay.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog(null, "Kindly choose the year, month, and
day!");
        } else {
            try {
                int cardId = Integer.parseInt(cardIdOfCreditCard);
                int cvcNumber = Integer.parseInt(CVCNumberCreditCard);
                double balanceAmount =
Double.parseDouble(balanceAmountCreditCard);
                double interestRate = Double.parseDouble(interestRateCreditCard);
                boolean cardIdFound = false;
                String expirationDate =
expirationDateComboBoxYear.getSelectedItem() + "-"
+expirationDateComboBoxMonth.getSelectedItem() + "-" +
expirationDateComboBoxDay.getSelectedItem();
                if (cardId <= 0) {
                    JOptionPane.showMessageDialog(null, "Card ID is invalid. Enter
a positive number only.\n");
                    return;
                } else if (balanceAmount <= 0) {
                    JOptionPane.showMessageDialog(null, "Balance Amount not
valid. Kindly input a positive number.\n");
                    return;
                } else if (cvcNumber <= 0) {
                    JOptionPane.showMessageDialog(null, "CVC number is invalid.
Kindly input a positive number.");
                    return;
                } else if (interestRate <= 0) {
                    JOptionPane.showMessageDialog(null, "Interest rate is invalid.
Enter a positive number only.");
                    return;
                } else {
                    for (int i = 0; i < bankCards.size(); i++) {
                        if (bankCards.get(i) instanceof CreditCard &&
bankCards.get(i).getCardId() == cardId) {
                            cardIdFound = true;
                            break;
                        }
                    }
                }
                if (cardIdFound) {
                    JOptionPane.showMessageDialog(null, "The card's unique
identifier has already been used or claimed by another person.!");
                } else {
                    CreditCard creditCard = new
CreditCard(cardId, clientNameOfCreditCard, issuerBankCreditCard, bankAccountCreditCard, ba
lanceAmount, cvcNumber, interestRate, expirationDate);
                    bankCards.add(creditCard);
                    JOptionPane.showMessageDialog(null, "Adding a credit card was
successful.");
                    clearEntry();
                }
            } catch (NumberFormatException ne) {
                JOptionPane.showMessageDialog(null, "Invalid input format. Please
enter a valid number for card ID, balance amount, CVC number, and interest rate.");
            }
        }
    });
}

```

```

        displayCreditCardBtn.addActionListener(e -> {
            new DisplayCreditCard(bankCards);
        });
    }
    public void clearEntry() {
        cardIdTextField.setText("");
        clientNameTextField.setText("");
        issuerBankTextField.setText("");
        bankAccountTextField.setText("");
        balanceAmountTextField.setText("");
        CVCNumberTextField.setText("");
        interestRateTextField.setText("");
        expirationDateComboBoxYear.setSelectedIndex(0);
        expirationDateComboBoxMonth.setSelectedIndex(0);
        expirationDateComboBoxDay.setSelectedIndex(0);
    }
}

```

## 8.6 Add Debit Card

```

package CourseWorkTwo.views;

import CourseWorkTwo.BankCard;
import CourseWorkTwo.Components. MyCustomBtn;
import CourseWorkTwo.Components.MyCustomLabel;
import CourseWorkTwo.Components.MyCustomTextField;
import CourseWorkTwo.DebitCard;

import javax.swing.*.*;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import java.awt.*.*;
import java.util.ArrayList;

import static java.awt.Color.*;

public class AddDebitCard extends JFrame {

    MyCustomLabel
    balanceAmountLabelDebitCard, cardIdLabelDebitCard, bankAccountLabelDebitCard, issuerBankLabelDebitCard, clientNameLabelDebitCard, pinNumberLabelDebitCard;
    MyCustomTextField
    balanceAmountTextFieldDebitCard, cardIdTextFieldDebitCard, bankAccountTextFieldDebitCard, issuerBankTextFieldDebitCard, clientNameTextFieldDebitCard, pinNumberTextFieldDebitCard;

    MyCustomBtn addDebitCardButton, clearBtnDebitCard, displayBtnDebitCard;

    public AddDebitCard(ArrayList<BankCard>bankCards) {
        super("Add Debit Card");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(700, 600);
        setLocationRelativeTo(null);
        setResizable(false);
        setVisible(true);

        //Creating panel for Debit card
        JPanel addDebitCardPanel = new JPanel();
        addDebitCardPanel.setBounds(80, 30, 100, 50);
        addDebitCardPanel.setBorder(new TitledBorder(new LineBorder(new

```

```

Color(0,0,0),10), "Add DebitCard", TitledBorder.LEFT, TitledBorder.TOP,null, new
Color(0,0,0)));
//      ,TitledBorder.CENTER,TitledBorder.TOP,null,new Color(172,216,230)));
//(.setBorder) creating TitledBorder,LineBorder and Color object to add
colorFull title with line border at Top and center
addDebitCardPanel.setBackground(lightGray);
addDebitCardPanel.setLayout(null);
add( addDebitCardPanel);

//creating label and textField for CardID Debit Card
cardIdLabelDebitCard =new MyCustomLabel("Card ID");
cardIdLabelDebitCard.setBounds(50,90,160,20);
addDebitCardPanel.add(cardIdLabelDebitCard);

cardIdTextFieldDebitCard = new MyCustomTextField("");
cardIdTextFieldDebitCard.setBounds(50,130,180,25);
addDebitCardPanel.add(cardIdTextFieldDebitCard);

//creating label and textField for PINNumber Debit Card
pinNumberLabelDebitCard =new MyCustomLabel("PIN Number");
pinNumberLabelDebitCard.setBounds(300,90,140,20);
addDebitCardPanel.add( pinNumberLabelDebitCard);

pinNumberTextFieldDebitCard = new MyCustomTextField("");
pinNumberTextFieldDebitCard.setBounds(300,130,180,25);
addDebitCardPanel.add(pinNumberTextFieldDebitCard);

//creating label and textField for Client Name Debit Card
clientNameLabelDebitCard =new MyCustomLabel("Client Name");
clientNameLabelDebitCard.setBounds(50,170,140,20);
addDebitCardPanel.add( clientNameLabelDebitCard);

clientNameTextFieldDebitCard = new MyCustomTextField("");
clientNameTextFieldDebitCard.setBounds(50,210,180,25);
addDebitCardPanel.add(clientNameTextFieldDebitCard);

//creating label and textField for Issuer Bank Debit Card
issuerBankLabelDebitCard =new MyCustomLabel("Issuer Bank");
issuerBankLabelDebitCard.setBounds(300,170,140,20);
addDebitCardPanel.add( issuerBankLabelDebitCard);

issuerBankTextFieldDebitCard = new MyCustomTextField("");
issuerBankTextFieldDebitCard.setBounds(300,210,180,25);
addDebitCardPanel.add(issuerBankTextFieldDebitCard);

//creating label and textField for Bank Account Debit Card
bankAccountLabelDebitCard =new MyCustomLabel("Bank Account");
bankAccountLabelDebitCard.setBounds(50,250,140,20);
addDebitCardPanel.add( bankAccountLabelDebitCard);

bankAccountTextFieldDebitCard = new MyCustomTextField("");
bankAccountTextFieldDebitCard.setBounds(50,290,180,25);
addDebitCardPanel.add(bankAccountTextFieldDebitCard);

//      creating label and textField for BalanceAmount Debit Card
balanceAmountLabelDebitCard =new MyCustomLabel("Balance Amount");
balanceAmountLabelDebitCard.setBounds(300,250,140,20);
addDebitCardPanel.add(balanceAmountLabelDebitCard);

balanceAmountTextFieldDebitCard = new MyCustomTextField("");
addDebitCardPanel.add(balanceAmountTextFieldDebitCard);

```

```

        balanceAmountTextFieldDebitCard.setBounds(300,290,180,25);

//<html><With>Withdraw From <br>Debit Card</body></html>
//Creating Add Debit Card button and clear button
addDebitCardButton = new MyCustomBtn("<html><With>Add To<br> Debit
Card</body></html>");
addDebitCardPanel.add(addDebitCardButton);
addDebitCardButton.setBounds(553,510,120,40);

//Clear Button for Debit Card
clearBtnDebitCard = new MyCustomBtn("Clear");
addDebitCardPanel.add(clearBtnDebitCard );
clearBtnDebitCard .setBounds(20,350,120,40);

//Display Button For Debit Card
displayBtnDebitCard = new MyCustomBtn("Display");
addDebitCardPanel.add(displayBtnDebitCard);
displayBtnDebitCard.setBounds(20,400,120,40);

addDebitCardButton.addActionListener(e ->{
    String balanceAmountOfDebitCard =
balanceAmountTextFieldDebitCard.getText();
    String cardIDOfDebitCard = cardIDTextFieldDebitCard.getText();
    String bankAccountOfDebitCard = bankAccountTextFieldDebitCard.getText();
    String issuerBankOfDebitCard = issuerBankTextFieldDebitCard.getText();
    String clientNameOfDebitCard = clientNameTextFieldDebitCard.getText();
    String pinNumberOfDebitCard = pinNumberTextFieldDebitCard.getText();

    if (balanceAmountOfDebitCard.equals("") || cardIDOfDebitCard.equals("") ||
bankAccountOfDebitCard.equals("") || issuerBankOfDebitCard.equals("") ||
clientNameOfDebitCard.equals("") || pinNumberOfDebitCard.equals("")) {
        JOptionPane.showMessageDialog(null, "Please check that all the
required fields are completed before proceeding");
    }else{
        try {
            double balanceAmount =
Double.parseDouble(balanceAmountOfDebitCard);
            int cardId = Integer.parseInt(cardIDOfDebitCard);
            int pinNumber = Integer.parseInt(pinNumberOfDebitCard);
            boolean cardIdFound = false;
            if (cardId <= 0) {
                JOptionPane.showMessageDialog(null, "Invalid card ID. Please
enter a positive integer.");
            } else if (balanceAmount <= 0) {
                JOptionPane.showMessageDialog(null, "Invalid balance amount.
Please enter a positive number.");
            } else if (pinNumber <= 0) {
                JOptionPane.showMessageDialog(null, "Invalid PIN number.
Please enter a positive integer.");
            } else {
                for (int i = 0; i < bankCards.size(); i++) {
                    if (bankCards.get(i) instanceof DebitCard &&
bankCards.get(i).getCardId() == cardId) {
                        cardIdFound = true;
                        break;
                    }
                }
            }
            if (cardIdFound) {
                JOptionPane.showMessageDialog(null, "Card identification
number has already been used by someone else!!");
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Please enter valid input");
        }
    }
});

```

```

        }else {
            DebitCard debitCard = new DebitCard(balanceAmount, cardId,
bankAccountOfDebitCard, issuerBankOfDebitCard, clientNameOfDebitCard, pinNumber);
            bankCards.add(debitCard);
            JOptionPane.showMessageDialog(null, "Debit Card added
successfully");
            clearEntry();
        }
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(null, "Invalid . Please enter a
valid number for balance amount, card ID, and PIN number.");
    }
}

});

displayBtnDebitCard.addActionListener(e -> {
    new DisplayDebitCard(bankCards);
});

}

public void clearEntry(){
    balanceAmountTextFieldDebitCard.setText("");
    cardIdTextFieldDebitCard.setText("");
    bankAccountTextFieldDebitCard.setText("");
    issuerBankTextFieldDebitCard.setText("");
    clientNameTextFieldDebitCard.setText("");
    pinNumberTextFieldDebitCard.setText("");
}

}
}

```

## 8.7 Cancel Credit Card

```

package CourseWorkTwo.views;

import CourseWorkTwo.BankCard;
import CourseWorkTwo.Components.MyCustomBtn;
import CourseWorkTwo.Components.MyCustomLabel;
import CourseWorkTwo.Components.MyCustomTextField;
import CourseWorkTwo.CreditCard;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;

```

```

import java.awt.*;
import java.util.ArrayList;

import static java.awt.Color.*;

public class CancelCreditCard extends JFrame {

    MyCustomLabel cancelCreditCardIdLabel;
    MyCustomTextField cancelCreditCardCardIdTextField;
    MyCustomBtn cancelCreditCardBtn , cancelCreditCardClearBtn;

    public CancelCreditCard(ArrayList<BankCard> bankCards) {
        super("Cancel Credit Card");
        setSize(700, 600);
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);
        setVisible(true);

        //Creating panel for cancel credit card
        JPanel cancelCreditCardPanel = new JPanel();
        cancelCreditCardPanel.setBounds(30,30,630,500);
        cancelCreditCardPanel.setBorder(new TitledBorder(new LineBorder(new
Color(173,216,230),10), "Cancel Credit Card", TitledBorder.LEFT,
TitledBorder.TOP,null, new Color(0,0,0)));
        //(.setBorder) creating TitledBorder,LineBorder and Color object to add
colorFull title with line border at Top and center
        cancelCreditCardPanel.setBackground(lightGray);
        cancelCreditCardPanel.setLayout(null);
        add(cancelCreditCardPanel);

        //Creating label and text field for card ID in set credit limit
        cancelCreditCardIdLabel =new MyCustomLabel("Card ID");
        cancelCreditCardIdLabel.setBounds(300,160,140,20);
        cancelCreditCardPanel.add(cancelCreditCardIdLabel);

        cancelCreditCardCardIdTextField = new MyCustomTextField("");
        cancelCreditCardPanel.add(cancelCreditCardCardIdTextField);
        cancelCreditCardCardIdTextField.setBounds(300,200,180,25);

        //Creating cancel Credit card button
        cancelCreditCardBtn = new MyCustomBtn("<html><With>Cancel
Credit<br>Limit</body></html> ");
        cancelCreditCardPanel.add(cancelCreditCardBtn);
        cancelCreditCardBtn.setBounds(553,510,120,40);

        //Clear Button for cancel credit card
        cancelCreditCardClearBtn = new MyCustomBtn("Clear");
        cancelCreditCardPanel.add(cancelCreditCardClearBtn );
        cancelCreditCardClearBtn .setBounds(300,250,120,40);

        cancelCreditCardClearBtn.addActionListener(e -> {
            clearEntry();
        });

        cancelCreditCardBtn.addActionListener(e ->{
            String cardId = cancelCreditCardCardIdTextField.getText();
            if(cardId.equals("")){
                JOptionPane.showMessageDialog(null,"Card Id field is empty.Please
Enter a Card Id");
            }else{

```



```

        try{
            int newCardId = Integer.parseInt(cardId);
            for (BankCard bankCard :bankCards) {
                if(bankCard instanceof CreditCard) {
                    if(bankCard.getCardId() ==newCardId) {
                        CreditCard creditCard = (CreditCard) bankCard;
                        creditCard.cancelCreditCard();
                        JOptionPane.showMessageDialog(null, "Credit card is
Successfully Cancelled:");
                        clearEntry();
                        return;
                    }
                }
            }
            JOptionPane.showMessageDialog(null,"No credit card found with the
entered card id.");
        } catch (NumberFormatException err) {
            JOptionPane.showMessageDialog(null,"Invalid!!Card Id,Please kindly
provide valid Card Id!");
        }
    }
    clearEntry();
    });
}

}
public void clearEntry () {
    cancelCreditCardCardIdTextField.setText("");
}
}
}

```

## 8.8 Display Credit Card

```

package CourseWorkTwo.views;

import CourseWorkTwo.BankCard;
import CourseWorkTwo.CreditCard;
import CourseWorkTwo.DebitCard;

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

public class DisplayCreditCard extends JFrame {
    public DisplayCreditCard(ArrayList<BankCard> bankCards) {
        super("Display Credit Cards");
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setSize(800, 700);
        setLocationRelativeTo(null);
        setResizable(false);
        setVisible(true);

        JPanel displayPanel = new JPanel();
        displayPanel.setLayout(new BoxLayout(displayPanel, BoxLayout.Y_AXIS));
    }
}

```

```

        String[] headersName = {"Card Id", "Client Name", "Issuer Bank", "Bank Account", "Card Type", "Balance Amount", "Interest Rate", "CVC Number", "Expiration Date"};
        ArrayList<CreditCard> creditCards = new ArrayList<>();
        for (BankCard bankCard : bankCards) {
            if (bankCard instanceof CreditCard) {
                creditCards.add((CreditCard) bankCard);
            }
        }
        String[][] detail = new String[creditCards.size()][9];
        for (int a = 0; a < creditCards.size(); a++) {
            detail[a][0] = String.valueOf(creditCards.get(a).getCardId());
            detail[a][1] = String.valueOf(creditCards.get(a).getClientName());
            detail[a][2] = String.valueOf(creditCards.get(a).getIssuerBank());
            detail[a][3] = String.valueOf(creditCards.get(a).getBankAccount());
            detail[a][4] = "Credit Card";
            detail[a][5] = String.valueOf(creditCards.get(a).getBalanceAmount());
            detail[a][6] = String.valueOf(creditCards.get(a).getInterestRate());
            detail[a][7] = String.valueOf(creditCards.get(a).getCvcNumber());
            detail[a][8] = String.valueOf(creditCards.get(a).getExpirationDate());
        }
        JTable displayTable = new JTable(detail, headersName);
        JScrollPane scrollPane = new JScrollPane(displayTable);
        displayPanel.add(scrollPane);
        add(displayPanel);
    }
}

```

## 8.9 Display Debit Card

```

package CourseWorkTwo.views;

import CourseWorkTwo.BankCard;
import CourseWorkTwo.DebitCard;

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

public class DisplayDebitCard extends JFrame {
    public DisplayDebitCard(ArrayList<BankCard> bankCards) {
        super("Display Debit Card");
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setSize(800, 700);
        setLocationRelativeTo(null);
        setResizable(false);
        setVisible(true);

        JPanel displaypanel = new JPanel();
        displaypanel.setLayout(new BoxLayout(displaypanel, BoxLayout.Y_AXIS));

        String[] headerNames = {"Card Id", "Bank Account", "Balance", "Client Name", "Card Type", "Pin", "Has Withdrawn"};
        ArrayList<DebitCard> debitCards = new ArrayList<>();
        for (BankCard bankCard : bankCards) {
            if (bankCard instanceof DebitCard) {
                debitCards.add((DebitCard) bankCard);
            }
        }
        String[][] detail = new String[debitCards.size()][7];
        for (int i = 0; i < debitCards.size(); i++) {

```

```

        detail[i][0] = String.valueOf(debitCards.get(i).getCardId());
        detail[i][1] = String.valueOf(debitCards.get(i).getBankAccount());
        detail[i][2] = String.valueOf(debitCards.get(i).getBalanceAmount());
        detail[i][3] = String.valueOf(debitCards.get(i).getClientName());
        detail[i][4] = "Debit Card";
        detail[i][5] = String.valueOf(debitCards.get(i).getPinNumber());
        detail[i][6] = String.valueOf(debitCards.get(i).getHasWithdrawn());
    }
    JTable displayTable = new JTable(detail, headerNames);
    JScrollPane scrollPane = new JScrollPane(displayTable);
    displaypanel.add(scrollPane);
    add(displaypanel);
}
}

```

## 8.10 main

```

package CourseWorkTwo.views;

import CourseWorkTwo.BankCard;
import CourseWorkTwo.Components.MyCustomBtn;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.util.ArrayList;

import static java.awt.Color.*;

public class Main extends JFrame {

    ArrayList<BankCard> bankCards = new ArrayList<>();

    MyCustomBtn
    addDebitCardButton, addCreditCardButton, withdrawalBtn, setCreditLimitBtn, cancelCreditCardBtn;

    public Main() {
        getContentPane().setBackground(white);
        setSize(800, 700);
        setLocationRelativeTo(null);
        setVisible(true);
        setLayout(null); //setting layout null to use our own layout(.setBounds)
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(30, 30, 630, 500);
        mainPanel.setBorder(new TitledBorder(new LineBorder(new Color(112, 128, 144),
10), "Hamro Bank Ltd", TitledBorder.LEFT, TitledBorder.TOP, null, new Color(0, 0, 0)));
        // , TitledBorder.CENTER, TitledBorder.TOP, null, new Color(172, 216,
230)));
        //(.setBorder) creating TitledBorder, LineBorder and Color object to add
colorFull title with line border at Top and center
        mainPanel.setBackground(lightGray);
        mainPanel.setSize(700, 600);
        mainPanel.setLayout(null);

        addDebitCardButton = new MyCustomBtn("Add DebitCard");
    }
}

```

```

        addDebitCardButton.setBounds(150,100,150,50);
        addCreditCardButton = new MyCustomBtn("Add CreditCard");
        addCreditCardButton.setBounds(350,100,150,50);
        withdrawalBtn = new MyCustomBtn("<html><With>Withdraw From <br>Debit
Card</body></html>");
        withdrawalBtn.setBounds(150,230,150,50);
        setCreditLimitBtn = new MyCustomBtn("Set Credit Limit");
        setCreditLimitBtn.setBounds(350,230,150,50);
        cancelCreditCardBtn = new MyCustomBtn("<html><With>Cancel<br>Credit
Card</body></html>");
        cancelCreditCardBtn.setBounds(245,360,150,50);

        addDebitCardButton.addActionListener(e ->{
            new AddDebitCard(bankCards);
        });

        addCreditCardButton.addActionListener(e ->{
            new AddCreditCard(bankCards);
        });

        withdrawalBtn.addActionListener(e ->{
            new WithdrawFromDebitCard(bankCards);
        });

        setCreditLimitBtn.addActionListener(e ->{
            new SetCreditLimit(bankCards);
        });

        cancelCreditCardBtn.addActionListener(e ->{
            new CancelCreditCard(bankCards);
        });

        mainPanel.add(addCreditCardButton);
        mainPanel.add(addDebitCardButton);
        mainPanel.add(withdrawalBtn);
        mainPanel.add(setCreditLimitBtn);
        mainPanel.add(cancelCreditCardBtn);

        add(mainPanel);
    }
}

```

## 8.11 Set Credit LI mit

```

package CourseWorkTwo.views;

import CourseWorkTwo.BankCard;
import CourseWorkTwo.Components.MyCustomBtn;
import CourseWorkTwo.Components.MyCustomLabel;
import CourseWorkTwo.Components.MyCustomTextField;
import CourseWorkTwo.CreditCard;

```

```

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.util.ArrayList;

import static java.awt.Color.*;

public class SetCreditLimit extends JFrame {
    MyCustomLabel
    cardIdLabelSetCreditLimit, creditLimitLabel, gracePeriodSetCreditLimitLabel;
    MyCustomTextField
    cardIdTextFieldSetCreditLimit, creditLimitTextField, gracePeriodSetCreditLimitTextField;
    MyCustomBtn setCreditLimitBtn , setCreditLimitClearBtn;
    public SetCreditLimit(ArrayList<BankCard> bankCards) {
        super("Withdraw from Debit Card");
        setSize(700, 600);
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);
        setVisible(true);

        //Creating panel for set credit limit
        JPanel setCreditLimitPanel = new JPanel();
        setCreditLimitPanel.setBounds(30,30,630,500);
        setCreditLimitPanel.setBorder(new TitledBorder(new LineBorder(new
Color(173,216,230),10), "Set Credit Limit", TitledBorder.LEFT, TitledBorder.TOP,null,
new Color(0,0,0)));
        //(.setBorder) creating TitledBorder,LineBorder and Color object to add
colorFull title with line border at Top and center
        setCreditLimitPanel.setBackground(lightGray);
        setCreditLimitPanel.setLayout(null);
        add( setCreditLimitPanel);

        //Creating label and text field for card ID in set credit limit
        cardIdLabelSetCreditLimit = new MyCustomLabel("Card ID");
        cardIdLabelSetCreditLimit.setBounds(300,115,140,20);
        setCreditLimitPanel.add( cardIdLabelSetCreditLimit);

        cardIdTextFieldSetCreditLimit = new MyCustomTextField("");
        setCreditLimitPanel.add(cardIdTextFieldSetCreditLimit);
        cardIdTextFieldSetCreditLimit.setBounds(300,150,180,25);

        //Creating label and text field for credit limit in set credit limit
        creditLimitLabel =new MyCustomLabel("Credit Limit");
        creditLimitLabel.setBounds(300,180,170,40);
        setCreditLimitPanel.add( creditLimitLabel);

        creditLimitTextField = new MyCustomTextField("");
        setCreditLimitPanel.add(creditLimitTextField);
        creditLimitTextField.setBounds(300,225,180,25);

        //Creating label and text field for grace period in set credit limit
        gracePeriodSetCreditLimitLabel =new MyCustomLabel("Grace Period");
        gracePeriodSetCreditLimitLabel.setBounds(300,255,170,40);
        setCreditLimitPanel.add(gracePeriodSetCreditLimitLabel);

        gracePeriodSetCreditLimitTextField = new MyCustomTextField("");
        setCreditLimitPanel.add(gracePeriodSetCreditLimitTextField);
    }
}

```

```

        gracePeriodSetCreditLimitTextField.setBounds(300,300,180,25);

        //Creating set Credit Limit button and clear button
        setCreditLimitBtn = new MyCustomBtn("Set Limit ");
        setCreditLimitPanel.add(setCreditLimitBtn);
        setCreditLimitBtn.setBounds(550,510,120,40);

        //Clear Button for set credit limit
        setCreditLimitClearBtn = new MyCustomBtn("Clear");
        setCreditLimitPanel.add(setCreditLimitClearBtn );
        setCreditLimitClearBtn .setBounds(300,350,120,40);

        setCreditLimitBtn.addActionListener(e ->{
            String cardId = cardIdTextFieldSetCreditLimit.getText();
            String creditLimit = creditLimitTextField.getText();
            String gracePeriod = gracePeriodSetCreditLimitTextField.getText();

            if(cardId.equals("") || creditLimit.equals("")) {
                JOptionPane.showMessageDialog(null, "Please ensure that all the
required fields are completed before proceeding!!");
                return;
            }
            try {
                int newCardId = Integer.parseInt(cardId);
                double newCreditLimit = Double.parseDouble(creditLimit);
                int newGracePeriod = Integer.parseInt(gracePeriod);

                if (newCardId <= 0 || newCreditLimit <= 0 || newGracePeriod <= 0) {
                    JOptionPane.showMessageDialog(null, "Invalid input detected for the
card ID, credit limit, and grace period fields!!");
                    return;
                }
                CreditCard creditCard = CreditCardFound(newCardId, bankCards);
                if (creditCard == null) {
                    JOptionPane.showMessageDialog(null, "The card you entered is not
recognized as a credit card!!");
                    return;
                }

                double balanceAmount = creditCard.getBalanceAmount();
                if(newCreditLimit>2.5*balanceAmount){
                    JOptionPane.showMessageDialog(null, "The credit card cannot be issued
more credit than 2.5 times its current balance, and the requested amount exceeds this
limit!!");
                    return;
                }

                creditCard.setCreditLimit(newCreditLimit,newGracePeriod);
                JOptionPane.showMessageDialog(null, "The system has updated your credit
limit successfully:) ");
                clearEntry();

            } catch (NumberFormatException err){
                JOptionPane.showMessageDialog(null, "Invalid input detected. Please
input only valid numbers for the card ID, credit limit, and grace period fields.!!");
            }

        });

    }
    private CreditCard CreditCardFound(int cardId, ArrayList<BankCard>bankCards) {
        for(BankCard bankCard : bankCards) {

```

```

        if(bankCard.getCardId()==cardId && bankCard instanceof CreditCard){
            return(CreditCard) bankCard;
        }
    }
    return null;
}

public void clearEntry(){
    cardIdTextFieldSetCreditLimit.setText("");
    creditLimitTextField.setText("");
    gracePeriodSetCreditLimitTextField.setText("");
}
}

```

## 8.12 With from debit card

```

package CourseWorkTwo.views;

import CourseWorkTwo.BankCard;
import CourseWorkTwo.Components.CustomComboBox;
import CourseWorkTwo.Components.MyCustomBtn;
import CourseWorkTwo.Components.MyCustomLabel;
import CourseWorkTwo.Components.MyCustomTextField;
import CourseWorkTwo.DebitCard;

import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.sql.Array;
import java.util.ArrayList;

import static java.awt.Color.*;

public class WithdrawFromDebitCard extends JFrame {
    MyCustomLabel cardIdLabelWithdrawal, withdrawalAmountLabel, dateOfWithdrawalLabel,
    pinNumberWithdrawalLabel;

    MyCustomTextField cardIdTextFieldWithdrawal, withdrawalAmountTextField,
    pinNumberWithdrawalTextField;

    CustomComboBox withdrawalComboBoxYear, withdrawalComboBoxMonth,
    withdrawalComboBoxDay;

    MyCustomBtn withdrawalBtn, clearBtnWithdrawal, displaybtn;

    public WithdrawFromDebitCard(ArrayList<BankCard> bankCards) {
        super("Withdraw from Debit Card");
        setSize(700, 600);
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);
        setVisible(true);

        //Creating panel for withdrawal from debit card
        JPanel withdrawalPanel = new JPanel();
        withdrawalPanel.setBounds(30, 30, 630, 500);
    }
}

```

```

        withdrawalPanel.setBorder(new TitledBorder(new LineBorder(new Color(173, 216,
230), 10), "Withdrawal From Debit Card", TitledBorder.LEFT, TitledBorder.TOP, null, new
Color(0,0,0)));
        //(.setBorder) creating TitledBorder,LineBorder and Color object to add
colorFull title with line border at Top and center
        withdrawalPanel.setBackground(lightGray);
        withdrawalPanel.setLayout(null);
        add(withdrawalPanel);

        //creating label and textField for CardId Withdrawal
        cardIdLabelWithdrawal = new MyCustomLabel("Card ID:");
        cardIdLabelWithdrawal.setBounds(50, 115, 140, 20);
        withdrawalPanel.add(cardIdLabelWithdrawal);

        cardIdTextFieldWithdrawal = new MyCustomTextField("");
        withdrawalPanel.add(cardIdTextFieldWithdrawal);
        cardIdTextFieldWithdrawal.setBounds(50, 165, 180, 25);

        //creating label and textField for PinNumber(Withdrawal from Debit Card)
        pinNumberWithdrawalLabel = new MyCustomLabel("PIN Number:");
        pinNumberWithdrawalLabel.setBounds(350, 115, 140, 20);
        withdrawalPanel.add(pinNumberWithdrawalLabel);
        pinNumberWithdrawalLabel.setForeground(white);
        pinNumberWithdrawalLabel.setFont(new Font("Tohama", Font.BOLD, 15));

        pinNumberWithdrawalTextField = new MyCustomTextField("");
        withdrawalPanel.add(pinNumberWithdrawalTextField);
        pinNumberWithdrawalTextField.setBounds(350, 165, 180, 25);

        //creating label and textField for withdrawal Amount
        withdrawalAmountLabel = new MyCustomLabel("Withdrawal Amount:");
        withdrawalAmountLabel.setBounds(50, 210, 170, 40);
        withdrawalPanel.add(withdrawalAmountLabel);

        withdrawalAmountTextField = new MyCustomTextField("");
        withdrawalPanel.add(withdrawalAmountTextField);
        withdrawalAmountTextField.setBounds(50, 270, 180, 25);

        //creating label and textField for date of withdrawal
        dateOfWithdrawalLabel = new MyCustomLabel("Date Of Withdrawal:"); //using
these tags for line break
        dateOfWithdrawalLabel.setBounds(350, 210, 170, 40);
        withdrawalPanel.add(dateOfWithdrawalLabel);

        //creating ComboBox(dropdown) for Expiration Date
        //ComboBox for years
        withdrawalComboBoxYear = new CustomComboBox(new String[]{
                "Year", "2016", "2017", "2018", "2019",
                "2020", "2021", "2022", "2023", "2024", "2025,", "2026", "2027",
"2028", "2029", "2030"

        });
        withdrawalComboBoxYear.setBounds(350, 270, 62, 25);
        withdrawalPanel.add(withdrawalComboBoxYear);

        //ComboBox for months
        withdrawalComboBoxMonth = new CustomComboBox(new String[]{
                "Month", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10",
"11", "12"
        });

```



```

        withdrawalComboBoxMonth.setBounds(415, 270, 72, 25);
        withdrawalPanel.add(withdrawalComboBoxMonth);

        //ComboBox for days
        withdrawalComboBoxDay = new CustomComboBox(new String[]{"Day",
            "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
            "12", "13", "14", "15", "16", "17", "18", "19", "20",
            "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"

        });
        withdrawalComboBoxDay.setBounds(490, 270, 55, 25);
        withdrawalPanel.add(withdrawalComboBoxDay);

        //Creating Withdrawal button and clear button for withdrawal from debit card
        withdrawalBtn = new MyCustomBtn("Withdrawal");
        withdrawalPanel.add(withdrawalBtn);
        withdrawalBtn.setBounds(550, 510, 120, 40);

        //Clear Button for withdrawal form debit card
        clearBtnWithdrawal = new MyCustomBtn("Clear");
        withdrawalPanel.add(clearBtnWithdrawal);
        clearBtnWithdrawal.setBounds(50, 340, 120, 40);

        //Display button for withdraw from debitCard
        displaybtn = new MyCustomBtn("Display");
        withdrawalPanel.add(displaybtn);
        displaybtn.setBounds(50, 390, 120, 40);

        displaybtn.addActionListener(e ->{
            new DisplayDebitCard(bankCards);
        });

        clearBtnWithdrawal.addActionListener(e ->{
            clearEntry();
        });

        withdrawalBtn.addActionListener(e -> {
            String cardId = cardIdTextFieldWithdrawal.getText();
            String withdrawalAmount = withdrawalAmountTextField.getText();
            String dateOfWithdrawal = withdrawalComboBoxYear.getSelectedItem() + " " +
            withdrawalComboBoxMonth.getSelectedItem() + " " +
            withdrawalComboBoxDay.getSelectedItem();
            String pinNumber = pinNumberWithdrawalTextField.getText();
            if (cardId.equals(" ") || withdrawalAmount.equals(" ") ||
            dateOfWithdrawal.equals(" ") || pinNumber.equals(" ") ||
            withdrawalComboBoxYear.getSelectedIndex() == 0 ||
            withdrawalComboBoxMonth.getSelectedIndex() == 0 ||
            withdrawalComboBoxDay.getSelectedIndex() == 0) {
                JOptionPane.showMessageDialog(null, "Please ensure that all the
                required fields are completed before proceeding!!");
            } else {
                try {
                    int withdrawalCardId = Integer.parseInt(cardId);
                    int withdrawal = Integer.parseInt(withdrawalAmount);

```

```

        int withdrawalPin = Integer.parseInt(pinNumber);
        boolean cardFound = false;
        DebitCard debitCard = null;

        if (withdrawalCardId <= 0) {
            JOptionPane.showMessageDialog(null, "Invalid card ID. Please
enter a positive integer.");
        } else {
            for (BankCard bankCard : bankCards) {
                if (bankCard instanceof DebitCard) {
                    debitCard = (DebitCard) bankCard;
                    if (debitCard.getCardId() == withdrawalCardId) {
                        cardFound = true;
                        break;
                    }
                }
            }
        }
        if (!cardFound) {
            JOptionPane.showMessageDialog(null, "The card you entered
cannot be found in the system.");
        } else if (withdrawal <= 0) {
            JOptionPane.showMessageDialog(null, "Invalid withdrawal
amount. Please enter a positive number.");
        } else {
            Withdraw(debitCard, withdrawalPin, withdrawal,
dateOfWithdrawal);
        }
    } catch (Exception err) {
        JOptionPane.showMessageDialog(null, "Invalid input format. Please
enter a valid number for card ID, withdrawal balance amount, and PIN number!");
    }
}

}

public void Withdraw(DebitCard debitCard, int pin, int withdrawal, String
dateOfWithdrawal) {
    if (debitCard.getPinNumber() == pin) {
        if (debitCard.getBalanceAmount() < (double)withdrawal) {
            JOptionPane.showMessageDialog((Component)null, "The account does not
have sufficient funds to process the requested transaction!!");
            return;
        }

        debitCard.Withdraw(withdrawal, dateOfWithdrawal, pin);
        JOptionPane.showMessageDialog((Component)null, "Your account has been
successfully debited with the withdrawn amount :)");
        this.clearEntry();
    } else {
        JOptionPane.showMessageDialog((Component)null, "The PIN you entered is
incorrect!!");
    }
}

public void clearEntry () {
    cardIdTextFieldWithdrawal.setText("");
    withdrawalAmountTextField.setText("");
    pinNumberWithdrawalTextField.setText("");
    withdrawalComboBoxYear.setSelectedIndex(0);
    withdrawalComboBoxMonth.setSelectedIndex(0);
}

```

```
        withdrawalComboBoxDay.setSelectedIndex(0);  
  
    }  
  
}
```