



 slington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2022 - 23 Autumn

Student Name: Atal Gyawali

London Met ID: 22067674

College ID: NP01CP4A220090

Group: C_5

Assignment Due Date: Wednesday, May 10, 2023

Assignment Submission Date: Tuesday, May 9, 2023

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

Contents

Introduction	1
Class Diagram.....	3
a) Bank Card class diagram	3
b) Debit Card class diagram	4
c) Credit Card class diagram.....	5
d) Whole Class Diagram	6
Pseudocode	9
a) Pseudocode of Bank Card Class	9
b) Pseudocode of Debit Card	12
c) Pseudocode of Credit Card	15
d) Pseudocode of BankGUI.....	18
Method Description (1 st Semester).....	29
a) Bank Card Class (Parent class)	29
b) Debit Card	30
c) Credit Card.....	31
Method Description (2 nd Semester)	32
TESTING (1 st Semester)	35
a) Test no 1 (Sem 1)	35
b) Test no 2 (Sem 1)	40
c) Test no 3 (Sem 1)	45
d) Test no 4	46
TESTING (2 nd Semester).....	48
a) Test no 1 (Sem 2).....	48
b) Test no 2 (Sem 2).....	49
c) Test no 2 (Sem 2)	54
Error Detection and Correction (1 st Semester)	56
a) Error 1 (Syntax Error).....	56
b) Error 2 (Semantic Error).....	57
c) Error 3 (Logical Error)	58
Error Detection and Correction (2nd Semester)	59

a) Error 1 (Syntax Error).....	59
b) Error 2 (Semantic Error)	61
c)Error 3 (Logical Error)	62
Conclusion	63
References.....	64
APPENDIX	65
a) Bank Card Class Code.....	65
b) Debit Card Class Code	68
c) Credit Card Class Code	71

Table of Figures

Figure 1 : Bank Card Class Diagram.....	3
Figure 2 : Debit Card Class Diagram.....	4
Figure 3 : Credit Card Class Diagram	5
Figure 4 : Whole Class Diagram	6
Figure 5 : Class diagram of BankGUI.....	7
Figure 6 : Whole class diagram including BankGUI	8
Figure 7 : Screenshot of assigning the data in Debit Card	36
Figure 8 : Screenshot of Inspecting Debit Card.....	37
Figure 9: Screenshot of entering the Withdraw Amount	38
Figure 10: Screenshot of re-inspecting Debit Card	39
Figure 11 : Screenshot of assigning the data in Credit Card	41
Figure 12 : Screenshot of Inspecting Credit Card	42
Figure 13 : Screenshot of putting Credit Limit and Grace Period	43
Figure 14 : Screenshot of re-inspecting Credit Card	44
Figure 15: Screenshot of inspecting again after cancelling the Credit Card	45
Figure 16 : Screenshot of details displayed in Debit Card.....	46
Figure 17: Screenshot of the details displayed in Credit Card.....	47
Figure 18 : Screen shot of GUI which was accessed through CMD.....	48
Figure 19 : Screen shot testing Add Debit Card button	50
Figure 20 : Screen shot of testing Display debit button after adding debit card.....	50
Figure 21 : Screen shot of testing withdraw button	51
Figure 22 : Screen shot of testing display debit button after withdrawing.....	51
Figure 23 : Screen shot of testing Add Credit button.....	52
Figure 24 : Screenshot of testing Set credit limit button.	52
Figure 25 : Screen shot of cancel credit card button.	53
Figure 26 : Screen shot of testing display credit button after cancelling credit card.	53
Figure 27 : Screen shot of testing add debit button without entering Card Id.	54
Figure 28 : Screen shot of testing add credit button without entering Card Id.	55
Figure 29 : Screen shot of testing add debit button when filling text field with string instead of number.....	55
Figure 30 : Screen shot of testing add credit button when filling text field with string instead of number.....	56
Figure 31: Screenshot of Syntax Error	56
Figure 32: Screenshot of Syntax Error Correction.....	57
Figure 33: Screenshot of Semantic Error	58
Figure 34: Screenshot of Semantic Error Correction.....	58
Figure 35: Screenshot of Logical Error.....	59
Figure 36: Screenshot of Logical Error Correction	59
Figure 37 : Syntax error (Semester 2)	60
Figure 38 : Syntax error correction (Semester 2)	60
Figure 39 : Semantic error (Semester 2)	61

Figure 40 : Semantic error correction (Semester 2)	61
Figure 41 : Logical error (Semester 2).....	62
Figure 42 : Logical error correction (Semester 2)	62

Table of Tables

Table 1 : Bank Card Class method description	29
Table 2 : Debit Card Class method description	30
Table 3 : Credit Card Class method description	31
Table 4 : Test 1 Inspecting Debit Card	35
Table 5 : Inspecting Credit Card.....	40
Table 6 : Inspecting Credit Card again after cancelling the Credit Card.....	45
Table 7 : Displaying the details of Debit and Credit Card	46
Table 8 : Testing no 1 of 2nd Semester.	48
Table 9 : Testing no 2 of 2nd Semester.	49
Table 10 : Testing no 3 of 2nd Semester	54

Introduction

The aim of this assignment was to add a class to the project that we developed in the previous semester to make a graphical user interface (GUI) for a system that stores details of Bank Card in an ArrayList. We were also supposed to add functionality to the GUI, the GUI should take input from the user and function accordingly. It should store Debit and Credit card details, display those details, set the credit limit, and cancel the credit card.

Java



Java is a popular, open source, free to use object-oriented programming language that is easy to learn and use. Java works on different platforms like Windows, Mac, Linux, etc. It's open-source and free to use. It is very similar to C++ and C# so it's easier for programmers to switch from Java to C++ or vice versa. Java is also secure, powerful, and fast. (Anon., n.d.).

Blue J



BlueJ is a software that allows us to develop Java programs easily and quickly. BlueJ has a smaller and simpler interface than other professional environments. It is very portable and runs on Windows, Mac, Linux, and other platforms which run Java. BlueJ is one of the best Java development environments for teaching. (Anon., n.d.) .

To complete this course work, we were supposed to use a software called BlueJ. BlueJ was very easy to learn because its interface was very friendly. It was too easy to detect any errors in BlueJ and the interface was very colourful which helped me to organize my code properly.

Microsoft Word



Microsoft word is a word processor software developed by Microsoft. It is one of the most popular word processor software in the world. It is used to create professional quality documents, letters, reports, resumes, etc. and allows you to edit or modify your new or existing document (Anon., n.d.).

This exact document file is created using Microsoft Word. I don't think I need to explain why I used Microsoft Word as my word processor software. It's easy to use, easy to learn, the most popular word processor.

Draw.io

Draw.io is an open-source diagramming software that allows the user to create



flowcharts, diagrams, class diagrams, etc. It is easy to learn and use. It has various functions like shape, text, automation, and connectors which we can use to create different types of diagrams. It is cross-platform which means it can be used in almost every device.

Draw.io was very helpful for creating class diagram of my program. It was very easy to use, and it had all the tools I needed to get the job done.

Class Diagram

Class diagram represents the types of objects residing in the system and the relationships between them. It shows the attributes, class, functions, and relationships.

It has class name, attributes, and functions which later helps in software development (Anon., n.d.).

To construct the class diagrams a software called draw.io was used.

a) Bank Card class diagram

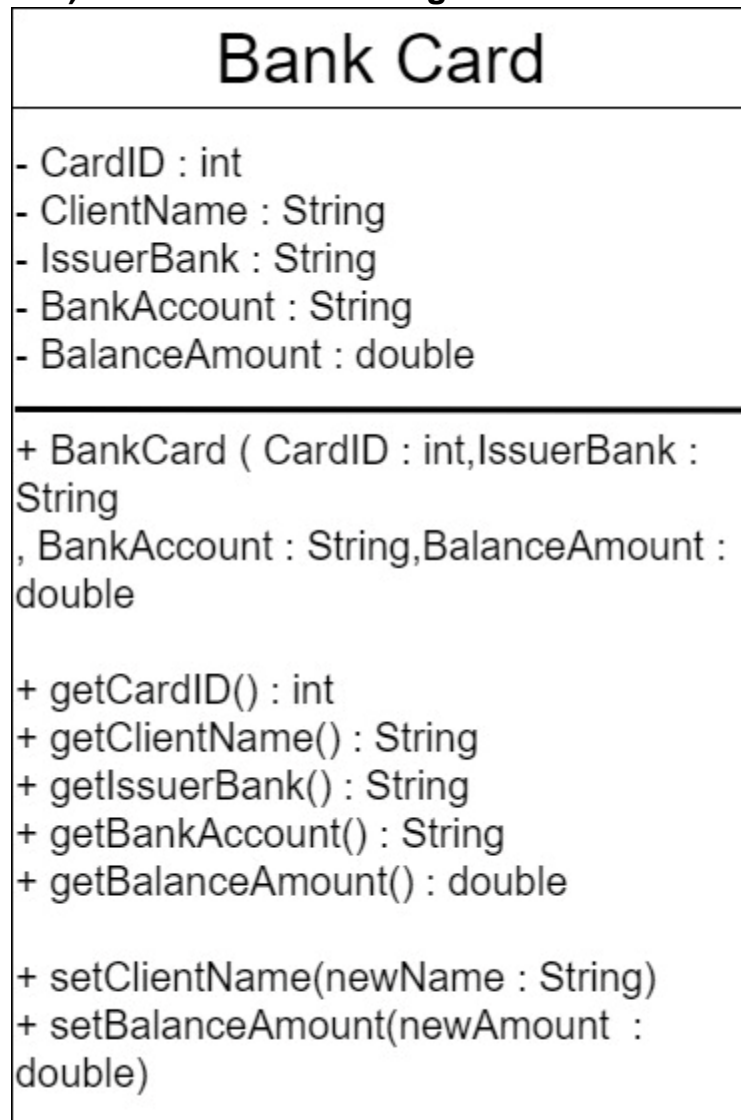


Figure 1 : Bank Card Class Diagram

b) Debit Card class diagram

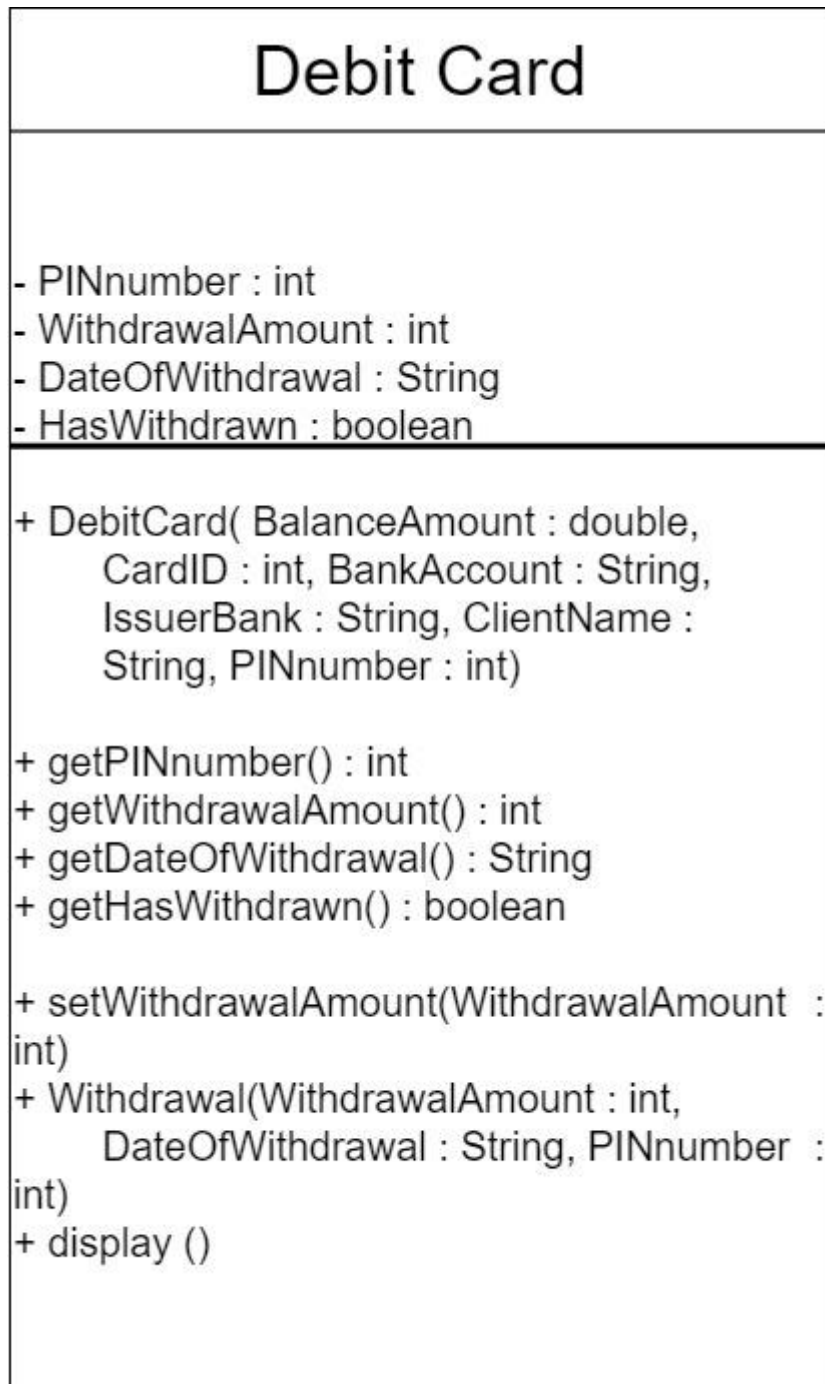


Figure 2 : Debit Card Class Diagram

c) Credit Card class diagram

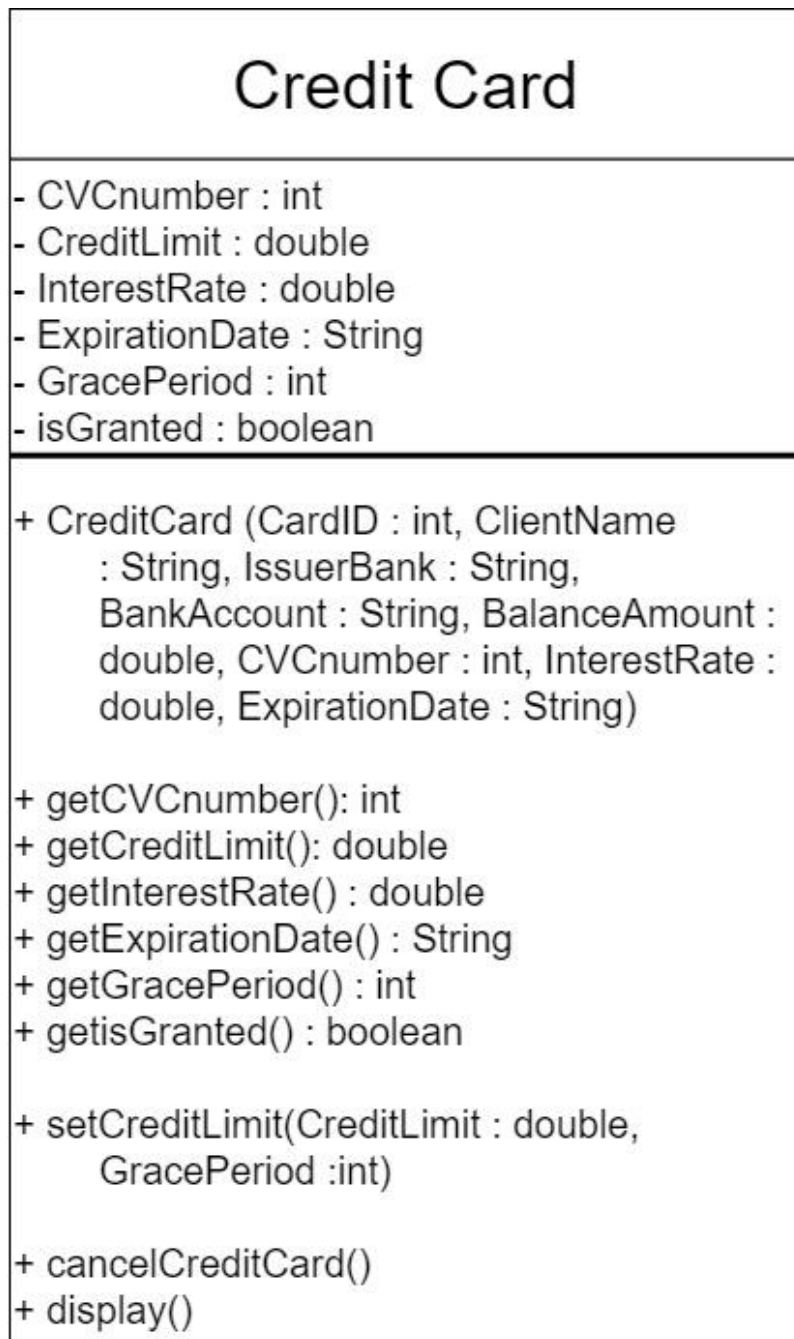


Figure 3 : Credit Card Class Diagram

d) Whole Class Diagram

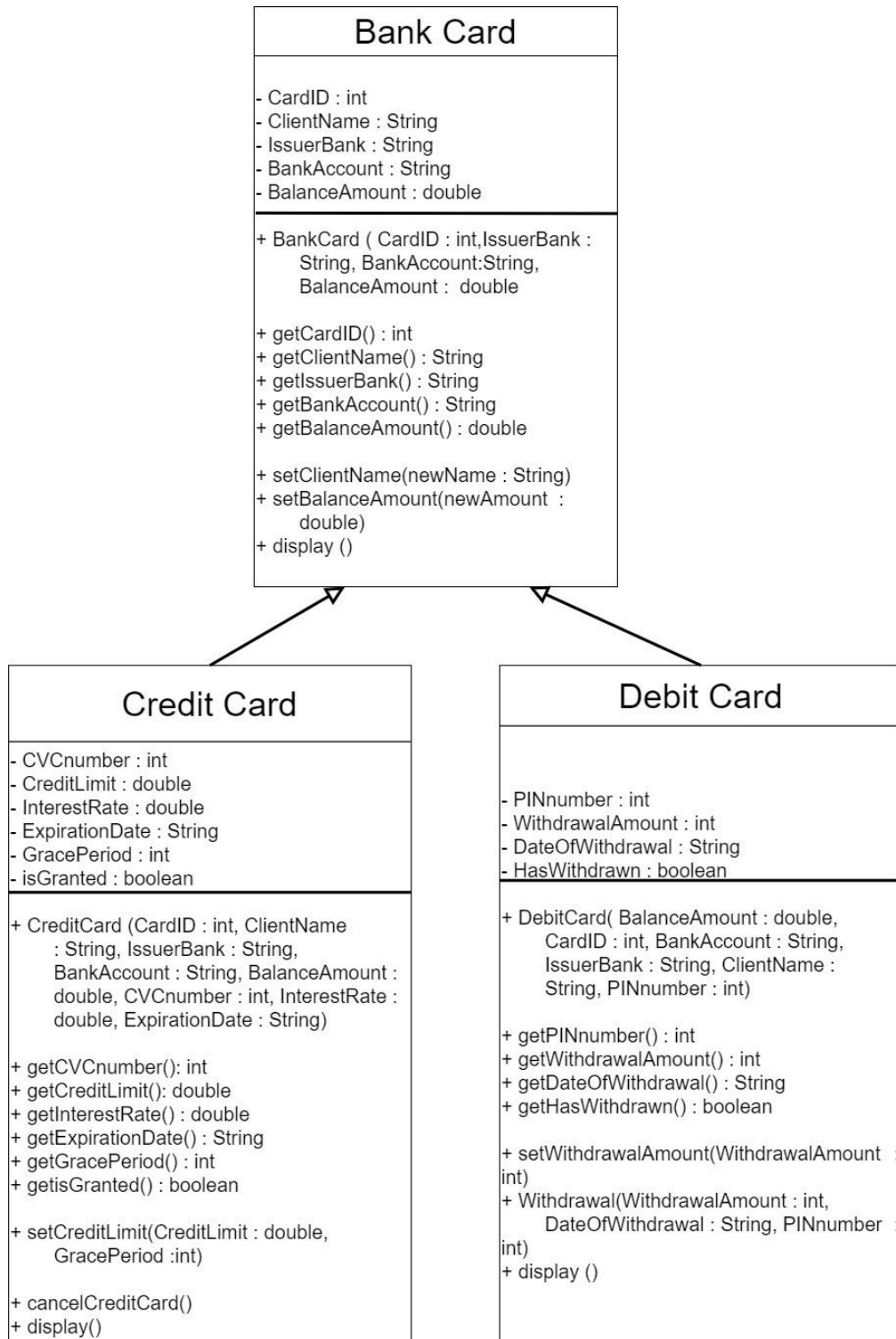


Figure 4 : Whole Class Diagram

e) BankGUI Class Diagram

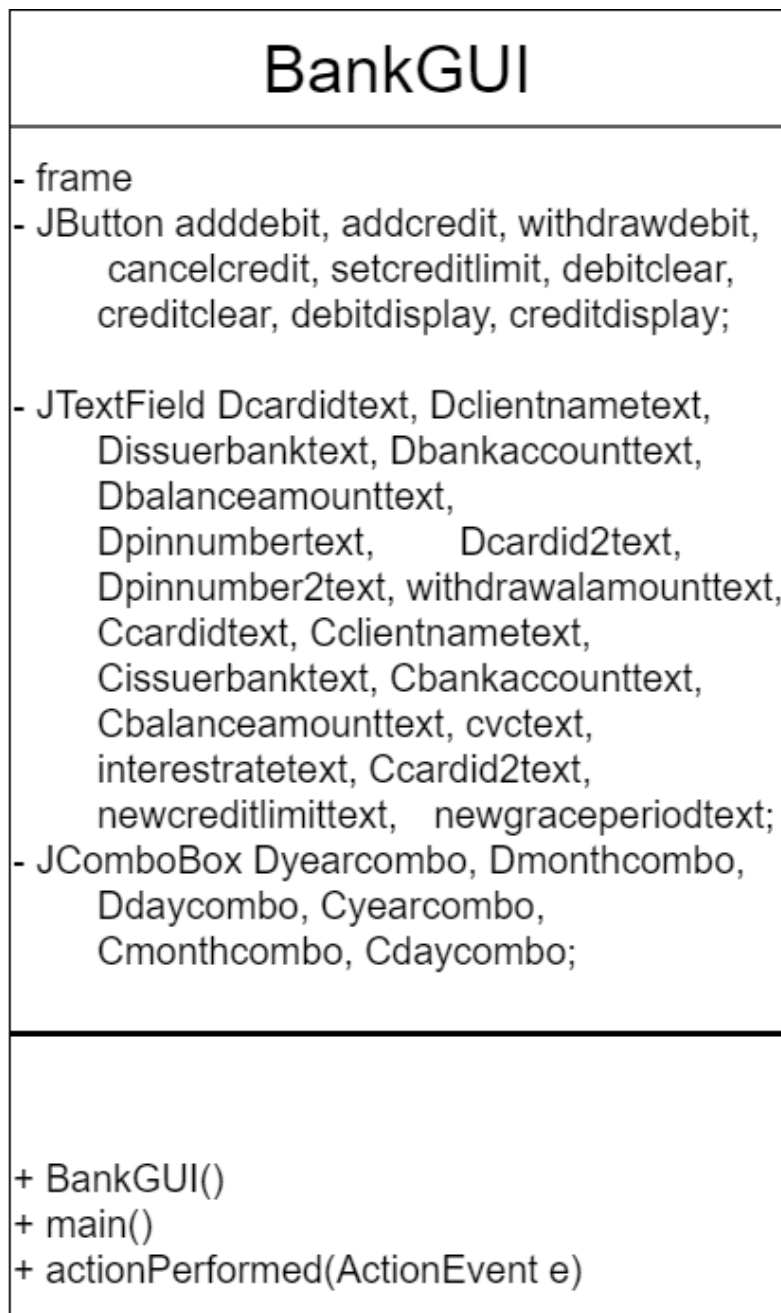


Figure 5 : Class diagram of BankGUI

f) Whole Class Diagram including BankGUI

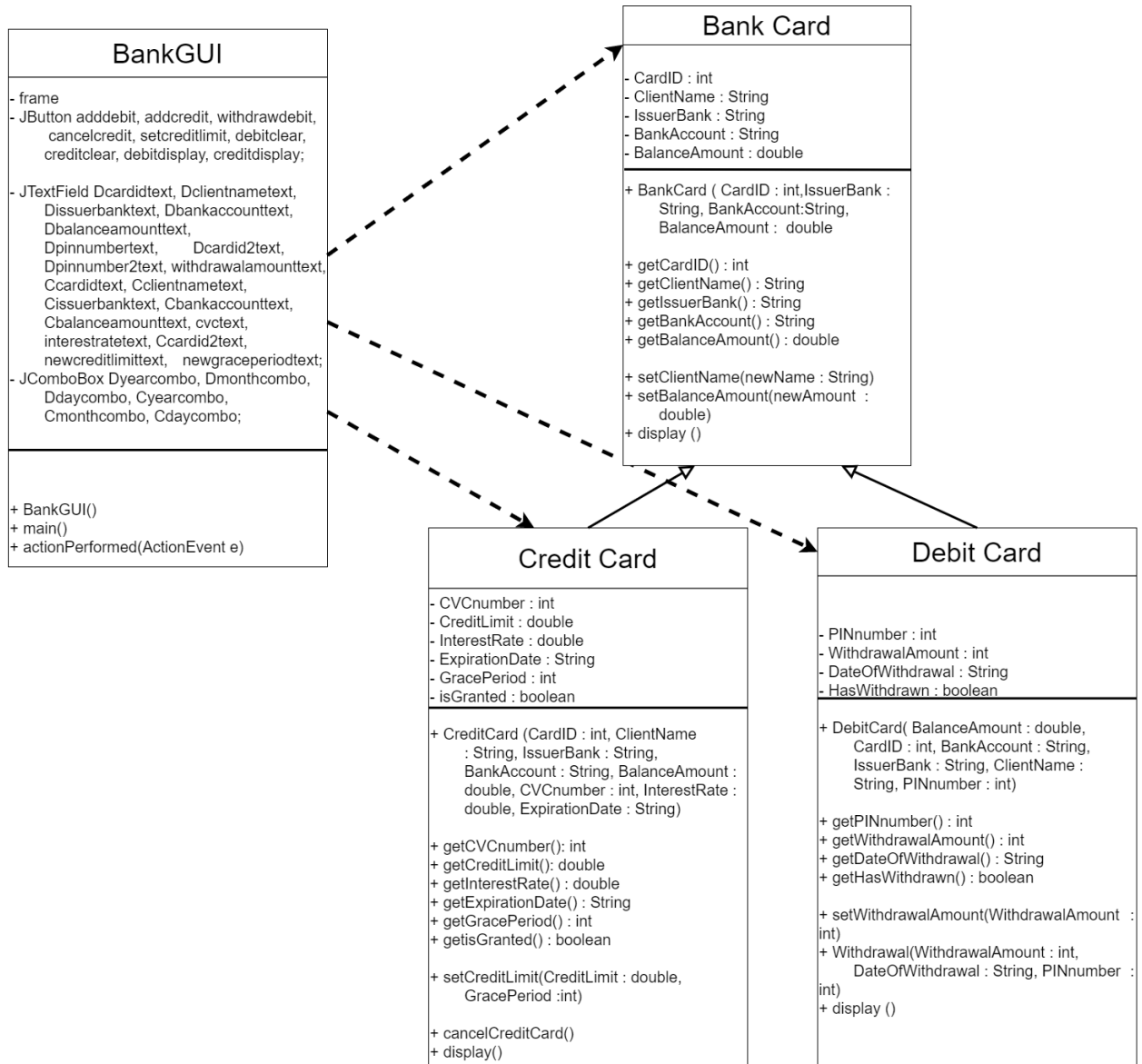


Figure 6 : Whole class diagram including BankGUI

Pseudocode

Pseudocode means fake code. It's an informal way of designing program where we don't have to obey any sets of rules. It is like a rough image of the actual program where we don't have to use semi-colons, curly braces, exact keywords, etc. We just must explain what we are doing in our own words (Anon., n.d.).

a) Pseudocode of Bank Card Class

Creating a public class BankCard

Creating five private attributes

Card id as number

Client Name as text

Issuer bank as text

Bank account as text

Balance amount as number

Creating a constructor called BankCard that has parameters: Card id, Client Name, Issuer bank, Bank account, Balance amount

Assigning attributes with parameter values

Setting the attribute client name empty

Putting an accessor method named getCardID which is an integer that returns CardID

Putting an accessor named getClientName which is a string of text that returns ClientName

Putting an accessor named getIssuerBank which is a string of text that returns IssuerBank

Putting an accessor named getBankAccount which is a string of text that returns BankAccount

Putting an accessor named getBalanceAmount which is an integer of text that returns BalanceAmount

Putting a setter named setClientName that sets newName which is a string as ClientName

Putting a setter named setBalanceAmount that sets newAmount which is an number as BalanceAmount

Putting a method named display

If client name is empty

Print "Client name is not given"

END IF

Else

Print "Card Id"

Print "Client Name"

Print "Issuer Bank"

Print "Bank Account"

Print "Balance Amount"

END ELSE

b) Pseudocode of Debit Card

Creating a public class Debit Card that is a sub-class of Bank Card class

Creating four private attributes

PIN number as number

Withdrawal amount as number

Date of withdrawal as text

Has withdrawal as true or false

Creating a constructor named DebitCard that has parameters: Balance amount, card id, Bank Account, Issuer bank, Client name, PIN number

Calling the parent class constructor with four parameters and a setter method named setclientname

Setting attribute PIN number equal to parameter PIN number

Putting Has withdrawn to false

Putting an accessor method named getPINnumber which is a number that returns PIN number

Putting an accessor method named getWithdrawalAmount which is an number that returns Withdrawal Amount

Putting an accessor method named getDateOfWithdrawal which is a String of text that returns Date of Withdrawal

Putting an accessor method named getHasWithdrawan which is either true or false that returns Has Withdrawal

Putting an accessor method named PINnumber which is a number that returns PIN number

Putting a setter that sets WithdrawalAmount which is a number to attribute Withdrawal amount

Creating a method name Withdraw that has parameter: Withdrawal Amount, Date of Withdrawal, PIN number

IF (Entered PIN is equal to attribute PIN number)

IF (Balance amount is greater than withdrawal amount)

deducting withdrawal amount from Balance amount

Setting attribute Withdrawal amount equal to parameter Withdrawal amount

Setting attribute Date of Withdrawal equal to parameter Date of Withdrawal

Setting the attribute hasWithdrawn to true

Print "Withdrawal Successful"

END IF

Else IF (Withdrawal Amount is greater than Balance Amount)

Print "Insufficient Balance"

END ELSE IF

Else IF (PIN number is wrong)

Print "Invalid PIN number"

END ELSE IF

Creating a display method that displays the details of the Debit Card

A call is made to the parent class to display card Id, client name, issuer bank, bank account, and Balance Amount

Print PIN number

IF Has withdrawn is true then

Print Withdrawal amount

Print Date of Withdrawal

Else

Print "No transaction yet"

Print Balance Amount

c) Pseudocode of Credit Card

Creating a public class Credit Card that is also a sub-class of Bank Card class

Creating six private attributes:

CVC number as an integer

Credit limit as double

Interest Rate as double

Expiration Date as String of character

Grace Period as integer

isGranted as Boolean (true or false)

Creating a constructor CreditCard that has 8 parameters: card Id, client name, issuerbank, bank account, BalanceAmount, CVC number, Interest rate, ExpirationDate

Calling the parent class constructor with four parameters and a setter method

Setting attribute CVC number equal to parameter CVC number

Setting attribute Interest rate equal to parameter Interest rate

Setting attribute Expiration Date equal to parameter Expiration Date

Setting attribute isGranted to false

Putting an accessor method named getCVCnumber which is a number that returns CVC number

Putting an accessor method named getCreditLimit which is a double that returns Credit Limit

Putting an accessor method named getInterestRate which is a double that returns Interest Rate

Putting an accessor method named getExpirationDate which is a String of character that returns Expiration Date

Putting an accessor method named getGracePeriod which is an integer that returns Grace Period

Putting an accessor method named getisGranted which is either true or false that returns is Granted

Creating a method named setCreditLimit that has parameter: Credit Limit and Grace Period

IF (Credit limit is less than or equal to 2.5 times Balance amount)

Setting attribute Credit Limit equal to parameter Credit Limit

Setting attribute Grace Period equal to parameter Grace Period

Setting isGranted attribute to true

END IF

Else

Print "Credit cannot be issued"

END ELSE

Creating a method cancelCreditCard

Setting the attribute CVCnumber to zero

Setting the attribute CreditLimit to zero

Setting the attribute GracePeriod to zero

Setting the attribute isGranted to zero

Creating a display method

IF (isGranted is true)

Calling the parent class display method

Print Credit Limit

Print Grace Period

END IF

Else

Print "Credit not Granted"

END ELSE

d) Pseudocode of BankGUI

Importing necessary classes, interfaces, and packages like:

Creating a public class BankGUI that implements Action Listener

Declaring Buttons, Text Fields, Combo Box and Array List

Creating a BankGUI method

Creating a new array list

Creating a Frame named frame

Creating 9 Buttons :

addebit

addcredit

withdrawdebit

cancelcredit

setcreditlimit

debitclear

creditclear

debitdisplay

creditdisplay

Creating 23 Lables

Creating 19 Text Fields:

Dcardidtext

Dclientnametext

Dissuerbanktext
Dbankaccounttext
Dbalanceamounttext
Dpinnumbertext
Dcardid2text
Dpinnumber2text
withdrawalamounttext
Ccardidtext
Cclientnametext
Cissuerbanktext
Cbankaccounttext
Cbalanceamounttext
cvctext
interestratetext
Ccardid2text
newcreditlimittext
newgraceperiodtext

Creating 6 combo box:

Dyearcombo
Dmonthcombo
Ddaycombo
Cyearcombo

Cmonthcombo

Cdaycombo

Setting bounds of buttons, labels, text fields and combo boxes

Adding action listener to buttons, text fields and combo boxes

Adding buttons, labels, text fields and combo boxes to the frame

Setting the frame size

Setting the frame layout

Setting default operation of the frame when close button is pressed

Setting frame Visibility

Setting resizabilty of the frame

BankGUI method ends here

Creating a main method

Creating a new frame that will be used to show message to the user

Creating an actionPerformed method for giving functionality

If (add debit button is pressed)

If (text fields are empty)

Show appropriate message

End IF

Else

Try

Entered card id, balance amount and pin
number should be numbers

Catch

Show appropriate message to the user

If (array list is empty)

Create object of debit card using constructor

Add to array list

Show message to the user

End If

Else

Loop through the array list

If card already exists

Don't add

Show message

End IF

Else

Add in array list

Show message

END else

End else

End else

End if

If (withdraw button is pressed)

If (Text fields are empty)

Show message

Else

Try

Entered card id , pin number and withdrawal
 amount should be in number format

Catch

Show appropriate message

Storing the entered data in variables

Looping through the array list

If (card exists in the array list)

If (the card Id and pin number matches)

If (balance amount is greater
 than withdraw)

Withdraw the entered
 amount

Show appropriate
 message

End If

Else

Show insufficient balance
 message

End else

End If

Else

Show pin number is wrong
message

End If

Else

Show card doesn't exist message

End else

If (Display debit button is pressed)

If (Text fields are empty)

Show appropriate message

End if

Else

Looping through the array list

If(card exists in the array list)

Display the card details

End if

End else

End if

If (add credit button is pressed)

If (text fields are empty)

Show appropriate message

End IF

Else

Try

Entered card id, balance amount, cvc number
and interest rate should be numbers

Catch

Show appropriate message to the user

If (array list is empty)

Create object of credit card using constructor

Add to array list

Show message to the user

End If

Else

Loop through the array list

If card already exists

Don't add

Show message

End IF

Else

Add in array list

Show message

END else

End else

End else

End if

If (Display credit button is pressed)

If (Text fields are empty)

Show appropriate message

End if

Else

Looping through the array list

If(card exists in the array list)

Display the card details

End if

End else

End if

If (set credit limit button is pressed)

If (Text fields are empty)

Show message

End if

Else

Try

Entered card id , new credit limit and grace period should be in number format

Catch

Show appropriate message

If (array list is empty)

Show appropriate message

End if

Else

Loop through the array list

If (card id exists in the array list)

If (credit limit is smaller than 2.5 times balance amount)

Set credit limit

Show success message

End if

End if

End else

End if

If(cancel credit card button is pressed)

If (Text fields are empty)

Show message

End if

Else

Try

Entered card id should be in number format

If (array list is empty)

Show appropriate message

End if

Else

Loop through the array list

IF (card exists in the array list)

IF (card id matches)

Show card cancelled
message

End if

End if

End else

Catch

Show card id should be number message

End else

End if

If (debit clear button is pressed)

Clear all the field in debit section in the GUI

End if

If (credit clear button is pressed)

Clear all the field in credit section in the GUI

End if

ActionPerformed method ends here

BankGUI class ends here

Method Description (1st Semester)

a) Bank Card Class (Parent class)

It is the main or parent class which consists of 9 methods that does the following things:

Table 1 : Bank Card Class method description

METHOD	FUNCTION
Bank Card (int CardId, String IssuerBank, String BankAccount, int BalanceAmount)	It's a constructor that accepts four parameter balance amount, card Id, bank account and issuer bank.
getCardId()	It gets card Id.
getClientName()	It gets Client Name.
getIssuerBank()	It gets Issuer Bank.
getBankAccount()	It gets Bank Account.
getBalanceAmount()	It gets Balance Amount.
setclientname (String newName)	It accepts new name as a parameter.
setbalanceamount (int newAmount)	It accepts new Amount as a parameter.
display()	It outputs the card Id, client name, issuer bank, bank account, balance amount and if the client name is not assigned it also outputs a message.

b) Debit Card

The debit card is a sub-class of bank card that has eight methods with the following functions:

Table 2 : Debit Card Class method description

Method	Function
DebitCard(int BalanceAmount, int CardId, String BankAccount, String IssuerBank,String ClientName, int PINnumber)	It's a constructor that accepts six parameters which are balance amount, card Id, bank account, issuer bank, client name, PIN number.
getPINnumber()	It gets PIN number.
getWithdrawalAmount()	It gets Withdrawal Amount.
getDateOfWithdrawal()	It gets Date of withdrawal.
getHasWithdrawn()	It gets Has Withdrawn.
setWithdrawalAmount(int WithdrawalAmount)	It sets the value of parameter withdrawal amount equal to attribute withdrawal amount
Withdraw(int WithdrawalAmount, String DateOfWithdrawal, int PINnumber)	It deducts money directly from the client account if the PIN is correct and sufficient amount is present.
display()	It displays the details of debit card

c) Credit Card

The credit card class is also a sub-class of the bank card class that has ten methods with the following functions:

Table 3 : Credit Card Class method description

Method	Function
CreditCard (int CardId, String ClientName, String IssuerBank, String BankAccount, int BalanceAmount, int CVCnumber, double InterestRate,String ExpirationDate)	It's a constructor that accepts parameter card Id, client name, issuer bank, bank account, Balance Amount, CVC number, Interest rate, Expiration Date.
getCVCnumber()	It gets CVC number.
getCreditLimit()	It gets Credit Limit.
getInterestRate()	It gets Interest Rate.
getExpirationDate()	It gets Expiration Date.
getGracePeriod()	It gets Grace Period.
getisGranted()	It gets Is Granted.
setCreditLimit(double CreditLimit, int GracePeriod)	It sets the Credit Limit.
cancelCreditCard()	It removes the client's credit card.
display()	It displays the details of credit card.

Method Description (2nd Semester)

BankGUI class

METHOD	FUNCTION
+ BankGUI()	This method has all the elements of the GUI. All the JLabels, JTextFields, JButtons, JComboBox , JFrame, are present in this method.
+main()	This main method calls the BankGUI method to display the created the GUI on the screen.
+actionPerformed(ActionEvent e)	<p>This method describes the function of each and every button in the GUI.</p> <p>For ex:</p> <ul style="list-style-type: none">- If (Add Debit) button is pressed without filling all the text field with suitable values, it shows an error message and if the user enters all the right values in all the text fields it checks if the array list is empty or not, if the array list is empty it adds the details provided by the user in the array list and if the array list is not empty it checks if the entered details already exists in the array list , if it already exists it again sends a message and if it doesn't it adds that details in the array list.- If (with draw from debit) button is pressed without filling all the required text field with suitable values, it shows an error message and if the user enters all the right values in all the text fields it checks if the entered Card Id and Pin number matches and if the withdraw amount

	<p>is smaller than balance amount , it withdraws the entered amount and shows a success message and if something doesn't match it shows an error message.</p> <ul style="list-style-type: none"> - If (Display Debit) button is pressed it displays the details of debit card present in the system and if there aren't any card in the system, it shows a message that says you haven't added any cards - If (Clear Debit) button is pressed it simply clears any text present in the text fields of debit card section - If (Add Credit) button is pressed without filling all the text field with suitable values, it shows an error message and if the user enters all the right values in all the text fields it checks if the array list is empty or not, if the array list is empty it adds the details provided by the user in the array list and if the array list is not empty it checks if the entered details already exists in the array list , if it already exists it again sends a message and if it doesn't it adds that details in the array list. - If (Display Credit) button is pressed, it shows all the details of credit card stored in the system. - If (Set credit limit) button is pressed after entering all the required text fields with suitable values and if the provided details matches with the details of the card present in the system it sets the credit limit according to the inputs given by the user and if the provided details doesn't
--	---

	<p>matches with the details of the card in the system it sends an error message to the user. It also shows appropriate error message to the user if the inputs are not suitable, wrong and if the system doesn't have any card details.</p> <ul style="list-style-type: none">- If (Cancel Credit) button is pressed after filling the text fields with suitable values it cancels the credit card by setting the card cvc number, credit limit, grace period to 0.- If (Clear Credit) button is pressed it simply clears any text present in the text fields of credit card section
--	---

TESTING (1st Semester)

a) Test no 1 (Sem 1)

Table 4 : Test 1 Inspecting Debit Card

Test no:	1
Objective:	To inspect the Debit card, withdraw the amount and re-inspect the Debit card.
Action:	<ul style="list-style-type: none">- The Debit card is called with the following arguments Balance amount = 25000 Card ID = 1 Bank Account = "55-49-26" Issuer Bank = "NRB" Client Name = "Hari" PIN number = 4567- Inspection of Debit Class- Void setWithdrawalAmount is called with the following arguments Withdrawal Amount = 2000- Re-inspection of Debit card
Expected Result:	The information about Debit card would be set including withdrawal amount.
Actual Result:	The information about Debit card was set including withdrawal amount.
Conclusion:	The test is successful.

Output Results

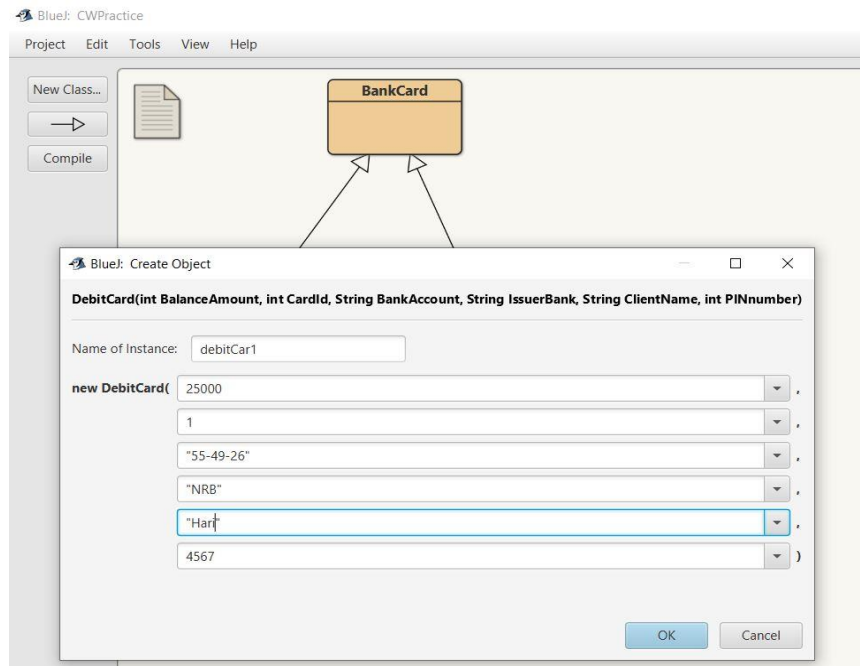


Figure 7 : Screenshot of assigning the data in Debit Card

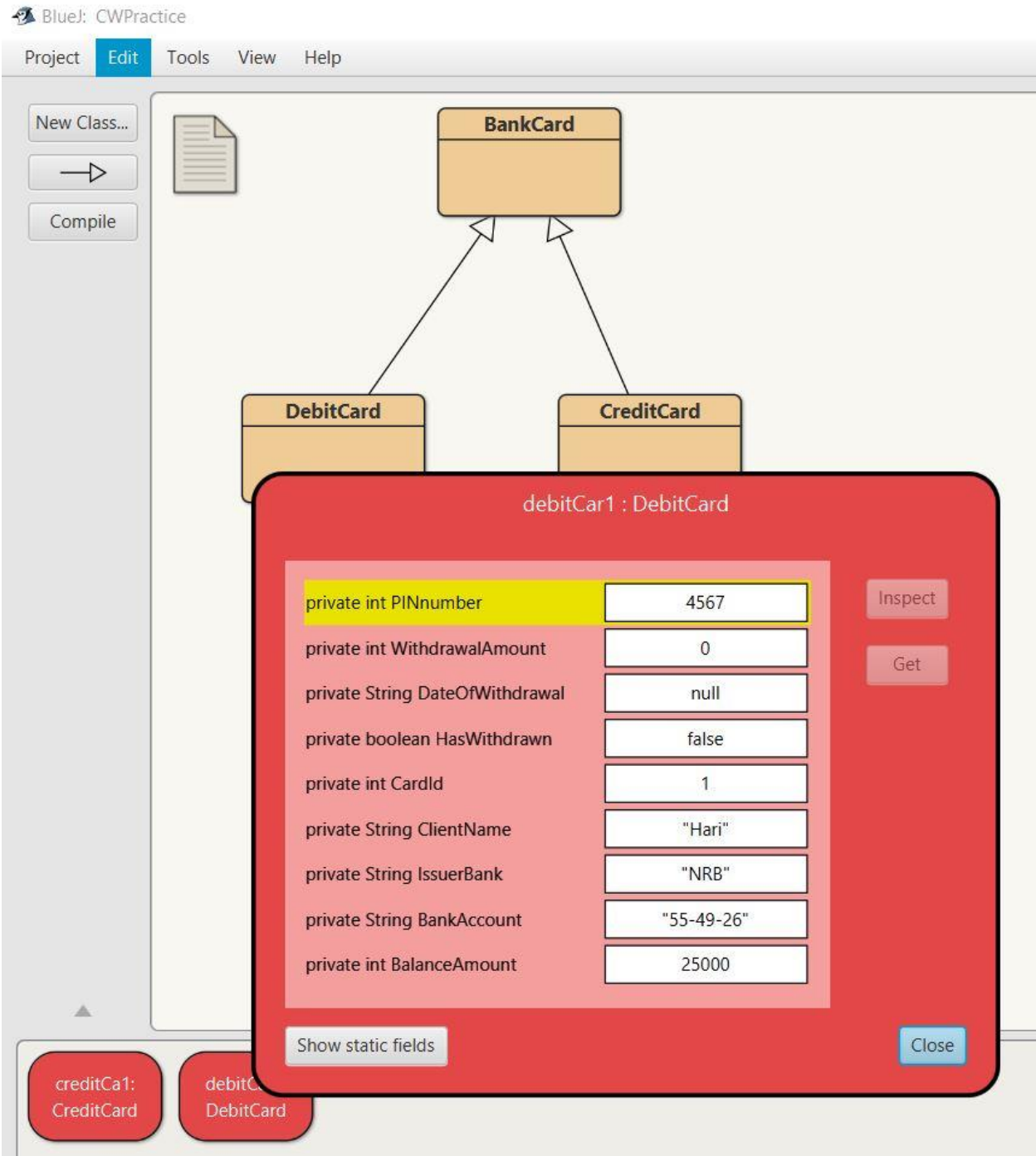


Figure 8 : Screenshot of Inspecting Debit Card

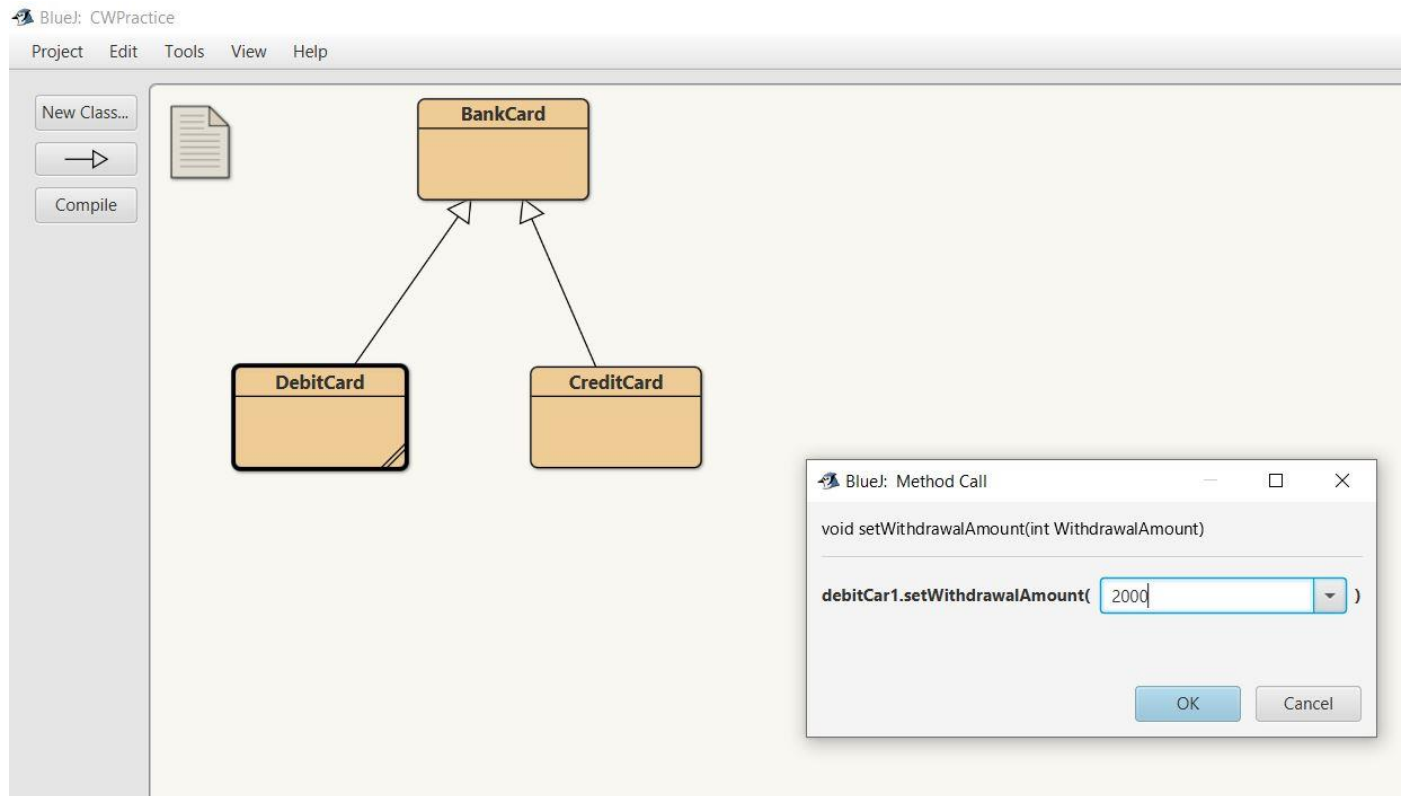


Figure 9: Screenshot of entering the Withdraw Amount

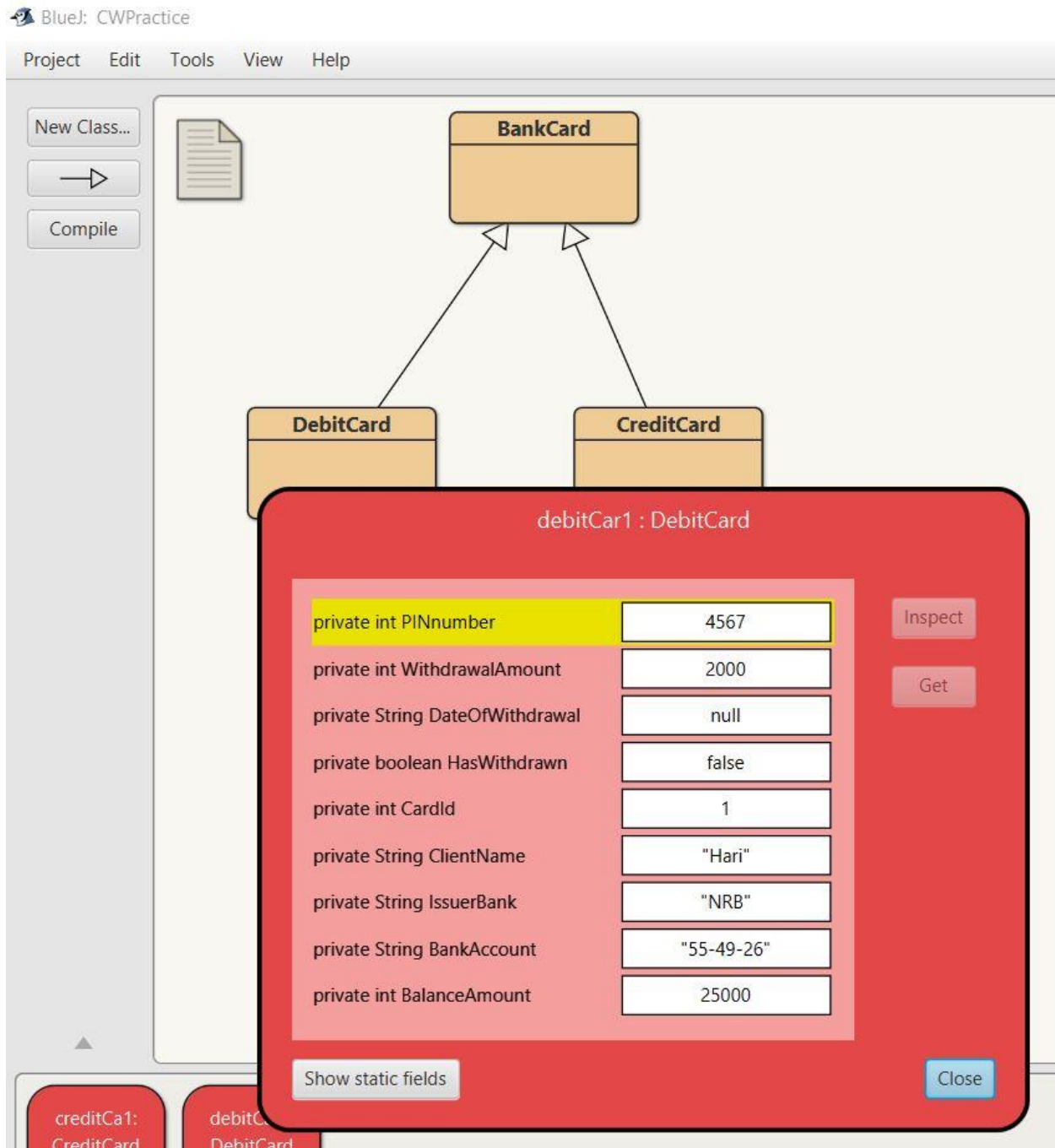


Figure 10: Screenshot of re-inspecting Debit Card

b) Test no 2 (Sem 1)

Table 5 : Inspecting Credit Card

Test no :	2
Objective:	To inspect the Credit Card class, set the credit limit and re-inspect the Credit Card class.
Action:	<ul style="list-style-type: none">- The credit card is called with the following arguments: CardId = 1 ClientName = "Hari" IssuerBank = "NRB" BankAccount = "55-49-26" BalanceAmount = 25000 CVCnumber = 123 InterestRate = 5 ExpirationDate = "2025-01-05"- Inspection of Credit Card class- void setCreditLimit is called with the following arguments: CreditLimit = 15000 GrancePeriod = 15- Re-inspection of Credit Card Class
Expected result:	The information about Credit card would be set including Credit Limit and Grace Period.
Actual result:	The information about Credit card was set including Credit Limit and Grace Period.
Conclusion:	The test is successful.

Output Results

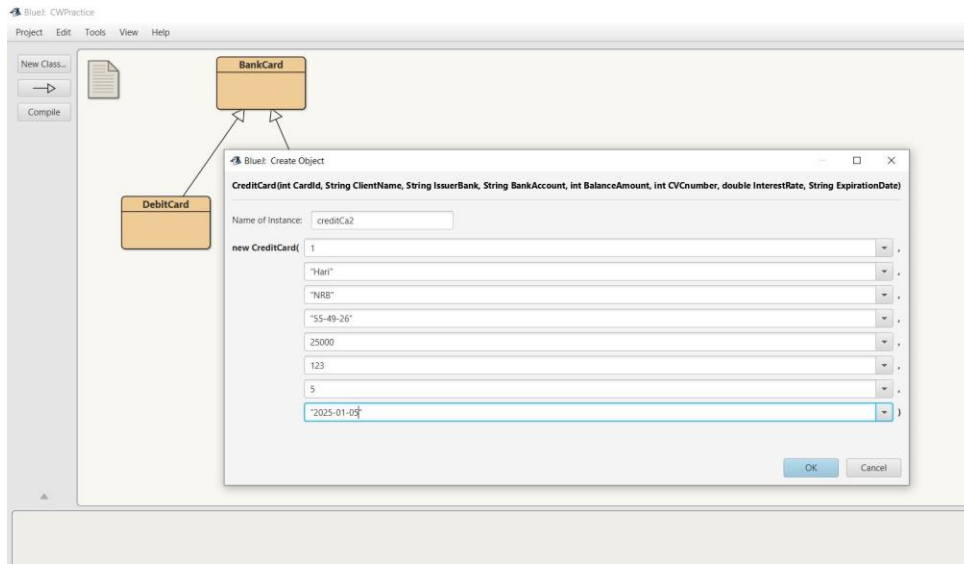


Figure 11 : Screenshot of assigning the data in Credit Card

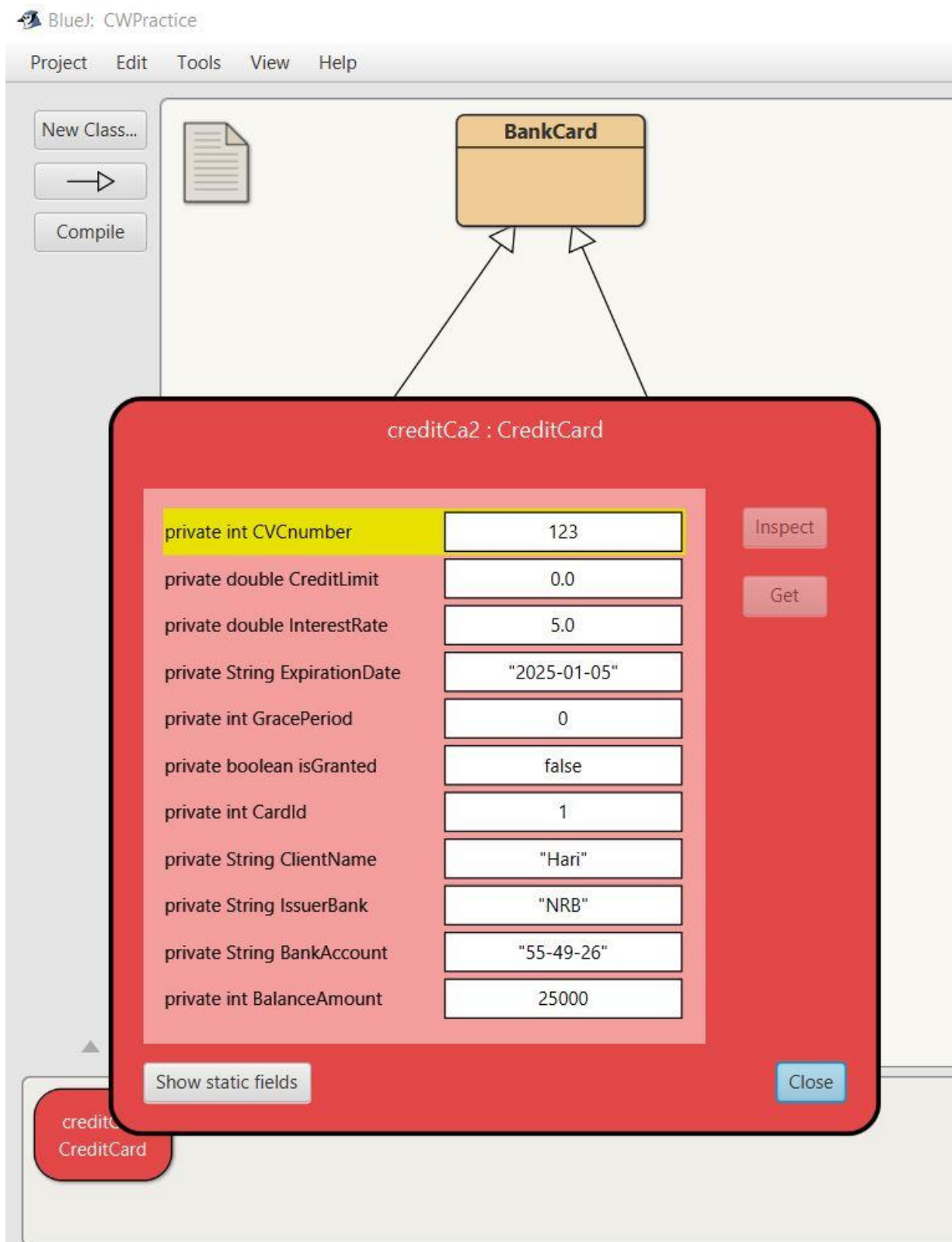


Figure 12 : Screenshot of Inspecting Credit Card

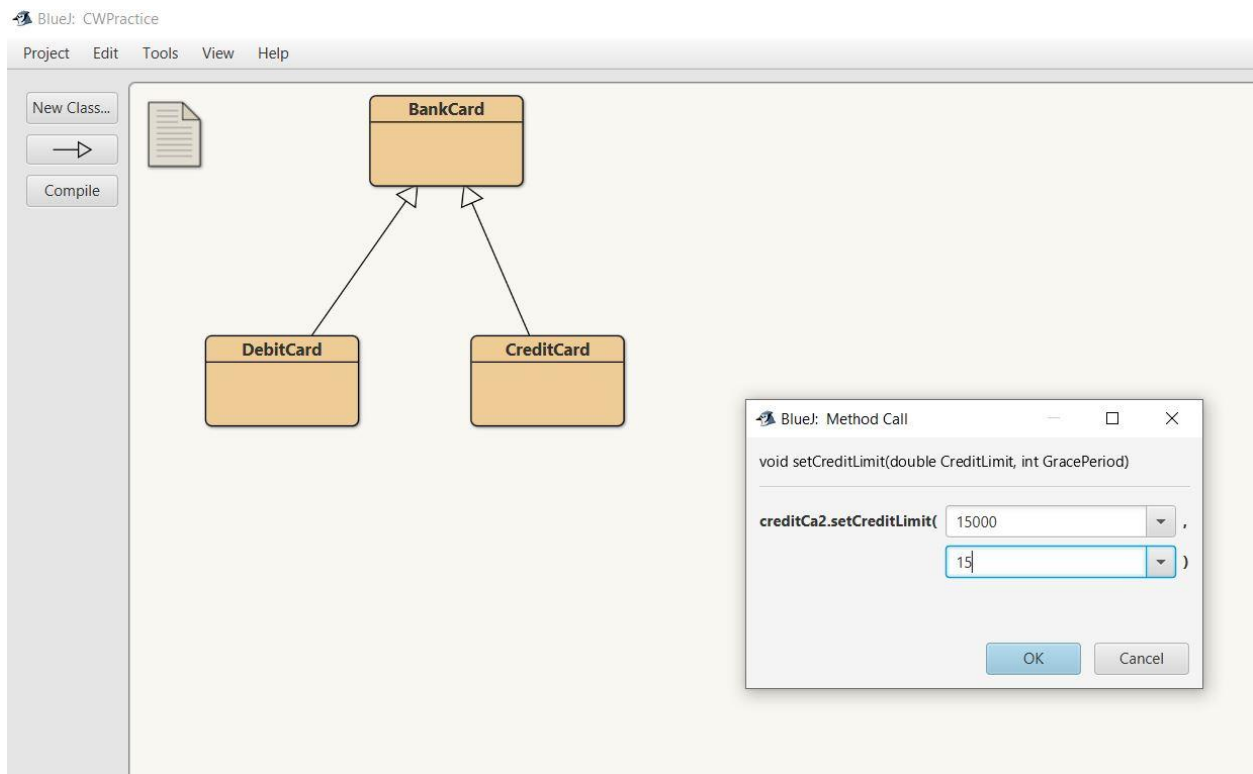


Figure 13 : Screenshot of putting Credit Limit and Grace Period

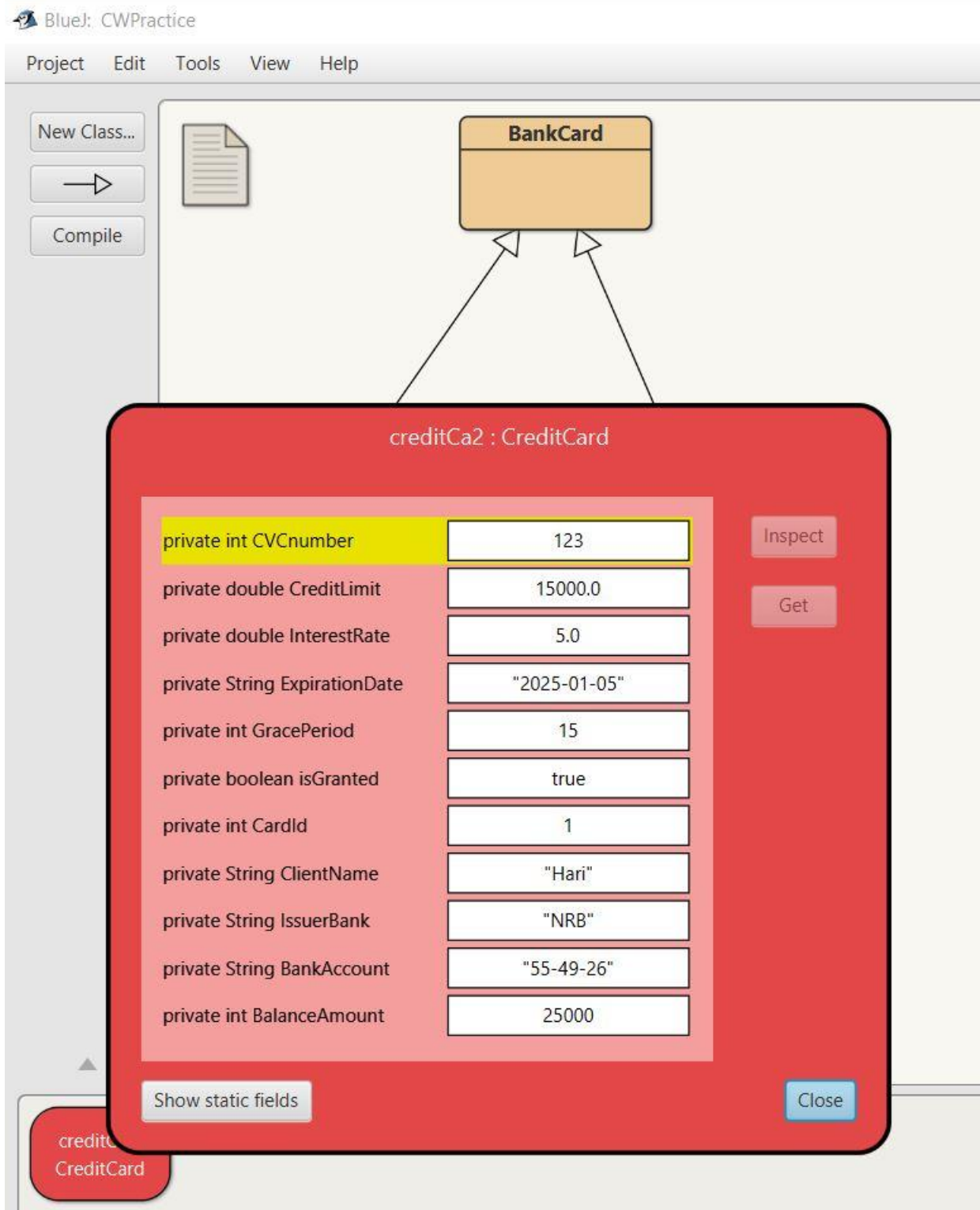


Figure 14 : Screenshot of re-inspecting Credit Card

c) Test no 3 (Sem 1)

Table 6 : Inspecting Credit Card again after cancelling the Credit Card

Test no :	3
Objective:	To inspect Credit Card class after cancelling the credit class.
Action:	- void cancelCreditCard is called
Expected result:	CVC number, Credit Limit and Grace Period should be zero.
Actual result:	CVC number, Credit Limit and Grace Period are zero.
Conclusion:	The test is successful.

Output Results

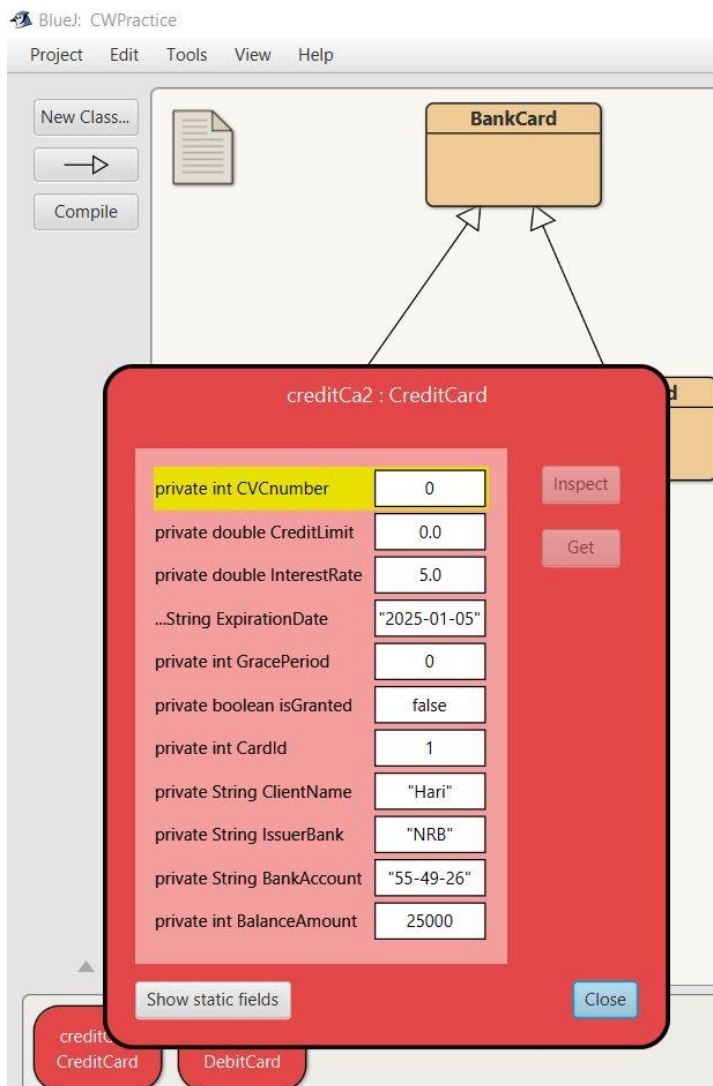


Figure 15: Screenshot of inspecting again after cancelling the Credit Card

d) Test no 4

Table 7 : Displaying the details of Debit and Credit Card

Test no :	4
Objective:	To display the details of Debit Card and Credit Card classes.
Action:	<ul style="list-style-type: none">- void display is called in Debit Card.- void display is called in Credit Card.
Expected result:	The details of both Debit Card and Credit Card should display.
Actual result:	The details of both Debit Card and Credit Card are displayed.
Conclusion:	The test is successful.

Output Results

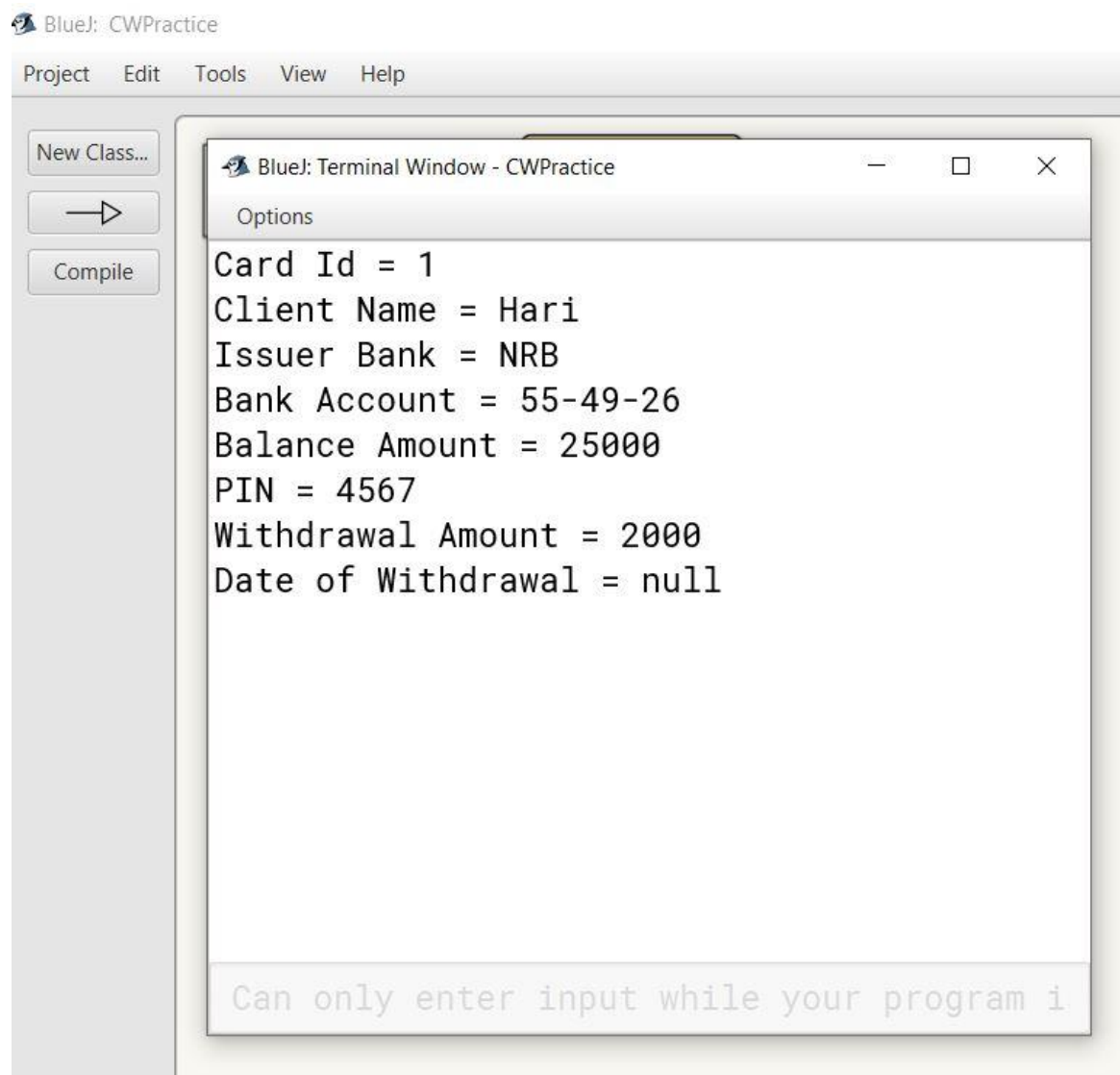


Figure 16 : Screenshot of details displayed in Debit Card

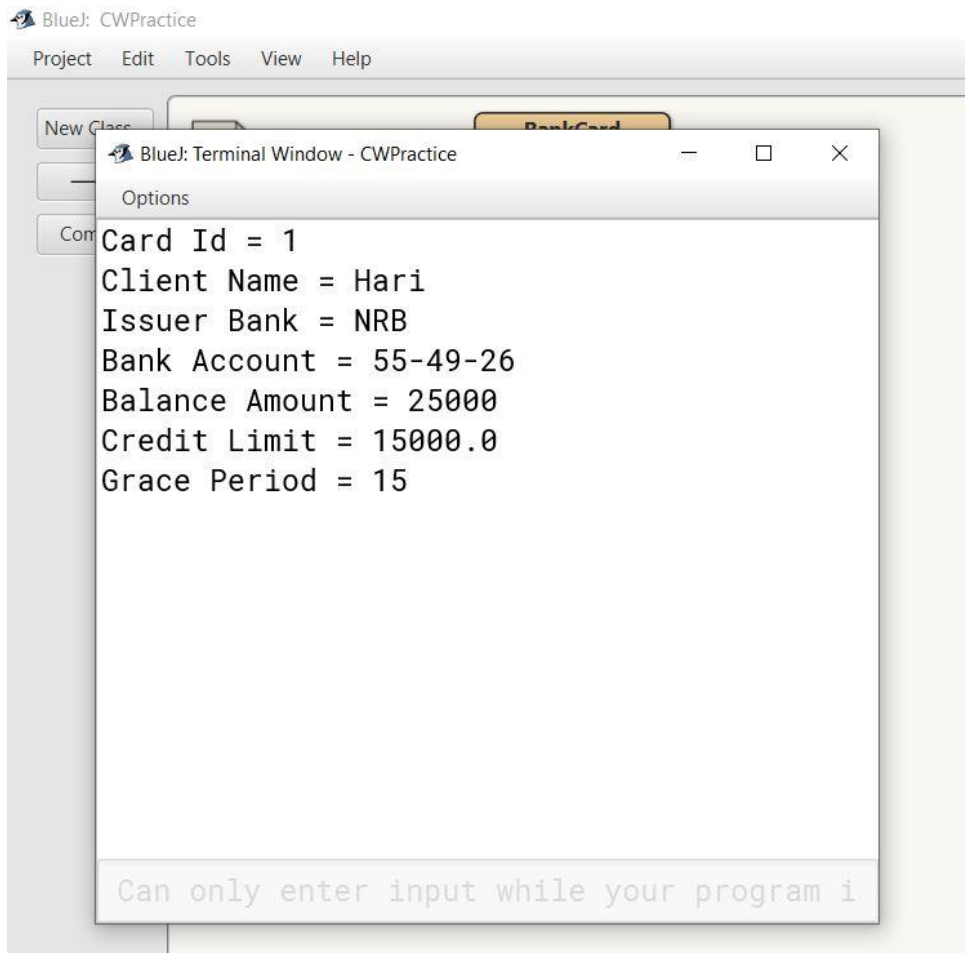


Figure 17: Screenshot of the details displayed in Credit Card

note: The screenshots were taken before properly naming my BlueJ project, that's why the project name is CWPractice in my screenshots.

TESTING (2nd Semester)

a) Test no 1 (Sem 2)

Table 8 : Testing no 1 of 2nd Semester.

Semester:	Second
Test no:	1
Objective:	Test that the program can be compiled and run using command prompt
Action:	<ul style="list-style-type: none">- Open CMD and enter the file location.- Type “javac <space> filename.java” to compile.- After solving any error and saving the program- Type “java <space> filename.java”
Expected result:	The BankGUI class should work and the frame containing the GUI should be shown in the screen.
Actual result:	The BankGUI class works and the frame containing the GUI is shown in the screen.
Conclusion:	The test is successful.

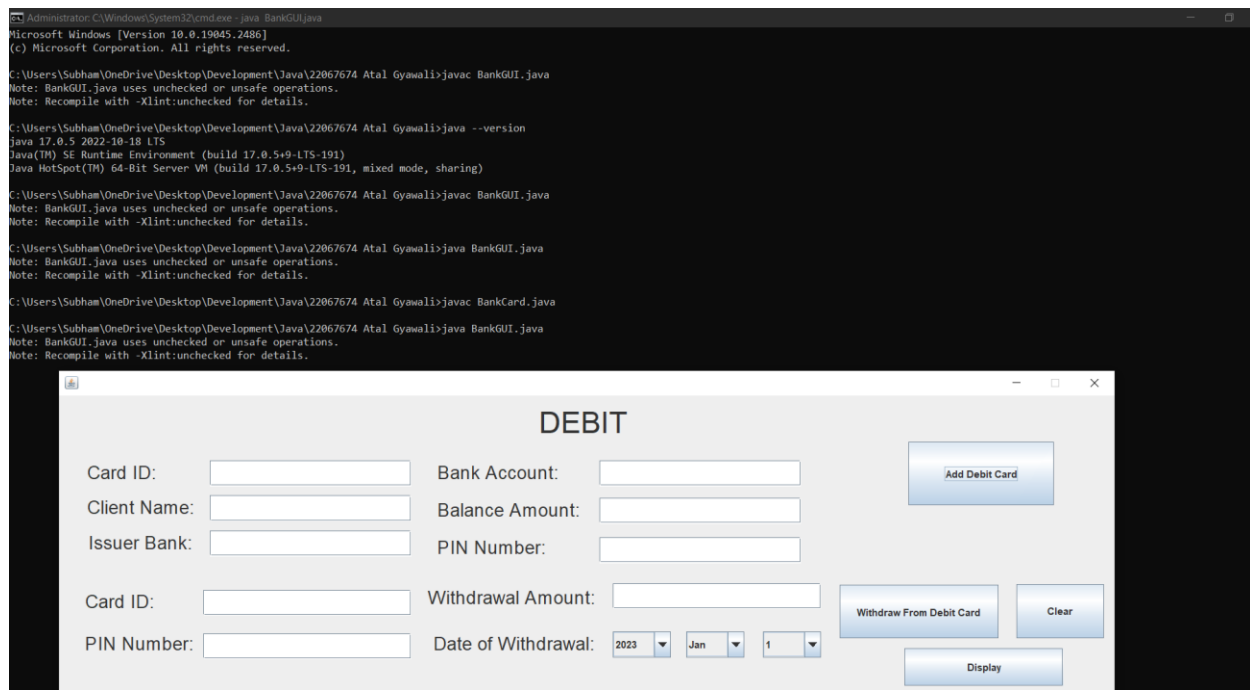


Figure 18 : Screen shot of GUI which was accessed through CMD.

b) Test no 2 (Sem 2)

Table 9 : Testing no 2 of 2nd Semester.

Semester:	Second
Test no:	2
Objective:	Testing the following buttons: a. Add DebitCard b. Add CreditCard c. Withdraw amount from Debit card. d. Set the credit limit. e. Remove the credit card
Action:	<ul style="list-style-type: none">- Clicking the Add DebitCard button after filling the text field in add debit card section.- Clicking the Add CreditCard button after filling the text field in add credit card section.- Clicking the Withdraw from Debit Card button after filling the text field in withdraw from debit card section.- Clicking the set the credit limit button after filling the text field in set the credit limit section.- Clicking the remove the credit card button
Expected result:	All the buttons should work properly, and appropriate message should be shown after clicking the buttons.
Actual result:	All the buttons work properly, and appropriate message are shown after clicking the buttons.
Conclusion:	The test is successful.

DEBIT

Card ID: Bank Account:

Client Name: Balance Amount:

Issuer Bank: PIN Number:

Card ID: Withdrawal Amount:

PIN Number: Date of Withdrawal:

CR

Card ID: Bank Account:

Client Name: Balance Amount:

Issuer Bank: CVC Number:

Card ID: New Credit Limit:

New Grace Period:

Message

Debit Card Added

OK

Figure 19 : Screen shot testing Add Debit Card button

DEBIT

Card ID: Bank Account:

Client Name: Balance Amount:

Issuer Bank: PIN Number:

Card ID: Withdrawal Amount:

PIN Number: Date of Withdrawal:

CR

Card ID: Bank Account:

Client Name: Balance Amount:

Issuer Bank: CVC Number:

Card ID: New Credit Limit:

New Grace Period:

Display

Debit Card.

Card ID: 1

Client Name: Atal

Issuer Bank: ing

Balance Amount: 10000

Bank Account: saving

Pin Number: 123

OK

Figure 20 : Screen shot of testing Display debit button after adding debit card

DEBIT

Card ID: 1 Bank Account: saving Add Debit Card

Client Name: Atal Balance Amount: 10000

Issuer Bank: ing PIN Number: 123

Card ID: 1 Withdrawal Amount: 500

PIN Number: 123 Date of Withdrawal: 2023 Jan 1 Withdraw From Debit Card Clear

Message: Successfully withdrawn from your accountCardID : 1
Pin Number :123
Withdraw Amount :500
Time :2023/Jan/1 OK

Card ID: Bank Account: Expiration Date: 2023 Jan 1 Display

Client Name: Balance Amount: Add Credit Card

Issuer Bank: CVC Number: Set Credit Card Clear

Card ID: New Credit Limit: Cancel Credit Card Display

New Grace Period:

Figure 21 : Screen shot of testing withdraw button

DEBIT

Card ID: Bank Account: Add Debit Card

Client Name: Balance Amount:

Issuer Bank: PIN Number:

Card ID: Withdrawal Amount:

PIN Number: Date of Withdrawal: 2023 Jan 1 Withdraw From Debit Card Clear

Display: Debit Card.
Card ID : 1
Client Name : Atal
Issuer Bank: ing
Balance Amount: 9500
Bank Account: saving
Pin Number: 123
Withdraw Amount: 500 OK

Card ID: Bank Account: Expiration Date: 2023 Jan 1 Display

Client Name: Balance Amount: Add Credit Card

Issuer Bank: CVC Number: Set Credit Card Clear

Card ID: New Credit Limit: Cancel Credit Card Display

New Grace Period:

Figure 22 : Screen shot of testing display debit button after withdrawing.

DEBIT

Card ID: Bank Account: Add Debit Card

Client Name: Balance Amount:

Issuer Bank: PIN Number:

Card ID: Withdrawal Amount: Withdraw From Debit Card Clear

PIN Number: Date of Withdrawal: 2023 Jan 1 Display

Card ID: 1 Bank Account: CUR st Rate: 3

Client Name: Atal Balance Amount: 5000 Expiration Date: 2023 Jan 1

Issuer Bank: ing CVC Number: 123 Add Credit Card

Card ID: New Credit Limit: Set Credit Card Clear

New Grace Period: Cancel Credit Card Display

Message
Credit Card Added
OK

Figure 23 : Screen shot of testing Add Credit button.

DEBIT

Card ID: Bank Account: Add Debit Card

Client Name: Balance Amount:

Issuer Bank: PIN Number:

Card ID: Withdrawal Amount: Withdraw From Debit Card Clear

PIN Number: Date of Withdrawal: 2023 Jan 1 Display

Card ID: 1 Bank Account: CUR st Rate: 3

Client Name: Atal Balance Amount: 5000 Expiration Date: 2023 Jan 1

Issuer Bank: ing CVC Number: 123 Add Credit Card

Card ID: 1 New Credit Limit: 1200 Set Credit Card Clear

New Grace Period: 3 Cancel Credit Card Display

Message
Credit Limit set
OK

Figure 24 : Screenshot of testing Set credit limit button.

The screenshot shows a web application titled "DEBIT" with two main sections. The top section contains input fields for Card ID, Bank Account, Client Name, Balance Amount, Issuer Bank, and PIN Number, along with an "Add Debit Card" button. The bottom section contains input fields for Card ID, Withdrawal Amount, PIN Number, Date of Withdrawal (with dropdowns for year, month, and day), and a "Withdraw From Debit Card" button. A "Display" button is also present. A message box is overlaid on the interface, displaying the text "Your credit card has been cancelled" with an "OK" button.

Figure 25 : Screen shot of cancel credit card button.

The screenshot shows the same "DEBIT" interface as Figure 25. A message box is overlaid, displaying the following details: "Credit Card. Card ID : 1, Client Name : Atal, Issuer Bank : ing, Balance Amount : 5000, Bank Account : current, CVC Number : 0, Interest Rate : 3.0, Expiration Date : 2023/Jan/1, Is Granted : false". The message box has an "OK" button.

Figure 26 : Screen shot of testing display credit button after cancelling credit card.

c) Test no 2 (Sem 2)

Table 10 : Testing no 3 of 2nd Semester.

Semester:	Second
Test no:	3
Objective:	Test that appropriate dialog boxes appear when unsuitable values are entered for the Card ID.
Action:	<ul style="list-style-type: none">- Clicking the add debit and add credit button without filling the text field of Card Id- Clicking the add debit and add credit button while filling the text filed with strings instead of integer.
Expected result:	All the buttons should work properly, and appropriate message should be shown after clicking the buttons.
Actual result:	All the buttons work properly, and appropriate message are shown after clicking the buttons.
Conclusion:	The test is successful.

The screenshot displays a banking application window titled "DEBIT". It features several input fields and buttons. The "Add Debit Card" button is highlighted. A "Message" dialog box is open, displaying the text "Please Input every field" and an "OK" button. The dialog box is positioned over the "Add Debit Card" button. The background interface includes fields for Card ID, Bank Account, Client Name, Issuer Bank, Balance Amount, PIN Number, Withdrawal Amount, Date of Withdrawal, and a "Display" button. There are also buttons for "Withdraw From Debit Card", "Clear", and "Add Credit Card".

Figure 27 : Screen shot of testing add debit button without entering Card Id.

DEBIT

Card ID: Bank Account:

Client Name: Balance Amount:

Issuer Bank: PIN Number:

Card ID: Withdrawal Amount:

PIN Number: Date of Withdrawal:

Message: Please Input every field

Buttons: Add Debit Card, Withdraw From Debit Card, Clear, Display

Figure 28 : Screen shot of testing add credit button without entering Card Id.

DEBIT

Card ID: Bank Account:

Client Name: Balance Amount:

Issuer Bank: PIN Number:

Card ID: Withdrawal Amount:

PIN Number: Date of Withdrawal:

Message: Card ID should be in number Format

Buttons: Add Debit Card, Withdraw From Debit Card, Clear, Display

Figure 29 : Screen shot of testing add debit button when filling text field with string instead of number.

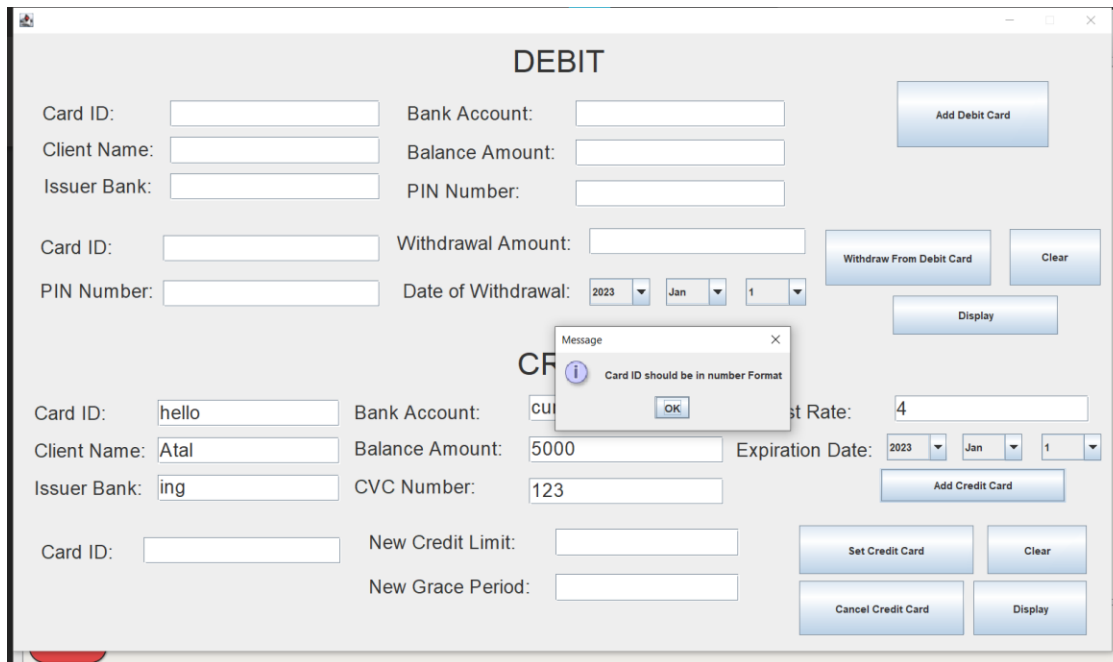


Figure 30 : Screen shot of testing add credit button when filling text field with string instead of number.

Error Detection and Correction (1st Semester)

a) Error 1 (Syntax Error)

This was the first error I found while writing my program. As we know every individual statement in java must be ended by a semicolon “ ; ”.

Here I did a simple mistake by not ending the statement by a semicolon.

```
//Constructor and Parameters
public BankCard (int CardId, String IssuerBank, String BankAccount, int BalanceAmount){
    this.CardId = CardId;
    this.IssuerBank = IssuerBank;
    this.BankAccount = BankAccount;
    this.BalanceAmount = BalanceAmount;
    this.ClientName = " "
}
// accessor method of each attributes
public int getCardId(){
    return CardId;
```

Figure 31: Screenshot of Syntax Error

I was able to correct this by simply adding a semicolon at the end of the statement.

```
//Constructor and Parameters
public BankCard (int CardId, String IssuerBank, String BankAccount, int BalanceAmount){
    this.CardId = CardId;
    this.IssuerBank = IssuerBank;
    this.BankAccount = BankAccount;
    this.BalanceAmount = BalanceAmount;
    this.ClientName = "";
}

// accessor method of each attributes
public int getCardId(){
    return CardId;
}
```

Figure 32: Screenshot of Syntax Error Correction

b) Error 2 (Semantic Error)

This was another error I found during programming. In this code I forgot to close the String (Card Id =) and the error occurred.

```

public void display(){
    if(ClientName.equals("")){
        System.out.println("Client name is not given");
    }
    else{
        System.out.println("Card Id = " + CardId);
        System.out.println("Client Name = " + ClientName);
        System.out.println("Issuer Bank = " + IssuerBank);
        System.out.println("Bank Account = " + BankAccount);
        System.out.println("Balance Amount = " + BalanceAmount);
    }
}

```

Figure 33: Screenshot of Semantic Error

I corrected the error by closing the String

```

public void display(){
    if(ClientName.equals("")){
        System.out.println("Client name is not given");
    }
    else{
        System.out.println("Card Id = " + CardId);
        System.out.println("Client Name = " + ClientName);
        System.out.println("Issuer Bank = " + IssuerBank);
        System.out.println("Bank Account = " + BankAccount);
        System.out.println("Balance Amount = " + BalanceAmount);
    }
}

```

Figure 34: Screenshot of Semantic Error Correction

c) Error 3 (Logical Error)

This was a simple logical error that I did while writing my code. The grace period was supposed to be zero but I wrote 10 instead of 0 which caused my program to work differently than intended.


```
public void cancelCreditCard(){  
    this.CVCnumber = 0;  
    this.CreditLimit = 0;  
    this.GracePeriod = 10;  
    this.isGranted = false;  
}
```

Figure 35: Screenshot of Logical Error

I corrected this error by simply changing the 10 to 0. Now, the program works perfectly

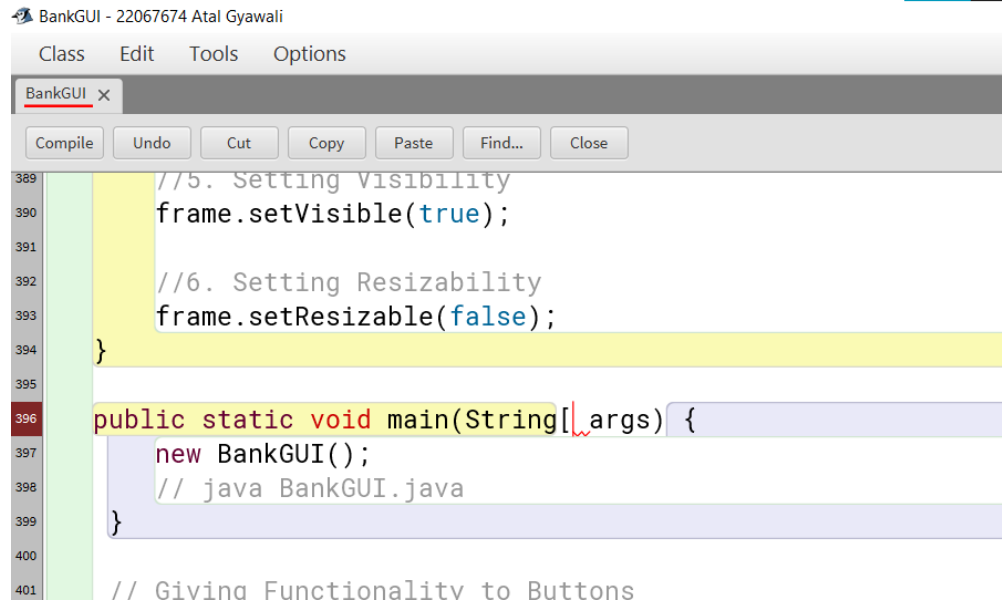
```
public void cancelCreditCard(){  
    this.CVCnumber = 0;  
    this.CreditLimit = 0;  
    this.GracePeriod = 0;  
    this.isGranted = false;  
}
```

Figure 36: Screenshot of Logical Error Correction

Error Detection and Correction (2nd Semester)

a) Error 1 (Syntax Error)

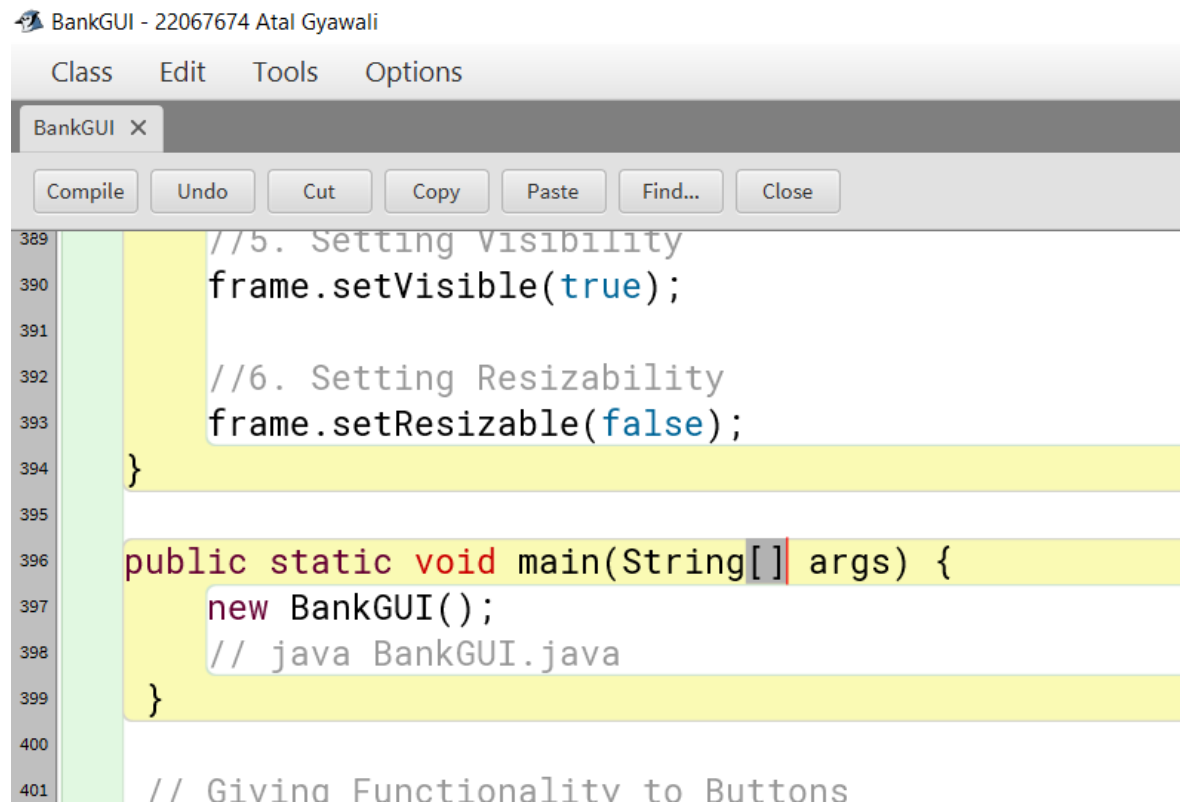
This was a simple syntax error I found while compiling my program. I forgot to close the big bracket "]" after String in my main method .



```
BankGUI - 22067674 Atal Gyawali
Class Edit Tools Options
BankGUI x
Compile Undo Cut Copy Paste Find... Close
389 //5. Setting Visibility
390 frame.setVisible(true);
391
392 //6. Setting Resizability
393 frame.setResizable(false);
394 }
395
396 public static void main(String[] args) {
397     new BankGUI();
398     // java BankGUI.java
399 }
400
401 // Giving Functionality to Buttons
```

Figure 37 : Syntax error (Semester 2)

I fixed this error by simply closing the bracket that was left open.



```
BankGUI - 22067674 Atal Gyawali
Class Edit Tools Options
BankGUI x
Compile Undo Cut Copy Paste Find... Close
389 //5. Setting Visibility
390 frame.setVisible(true);
391
392 //6. Setting Resizability
393 frame.setResizable(false);
394 }
395
396 public static void main(String[] args) {
397     new BankGUI();
398     // java BankGUI.java
399 }
400
401 // Giving Functionality to Buttons
```

Figure 38 : Syntax error correction (Semester 2)

b) Error 2 (Semantic Error)

Here I forgot to convert the Dcardidtext into integer. So, when I entered string value instead of number in the Card Id text field, the program was still accepting the string value which should not happen.

```
JOptionPane.showMessageDialog( msgframe, "Please input every  
}  
else {  
    // Sending user a message if he/she enters CardID in any form  
    try{  
        Dcardidtext.getText();  
    }  
    catch(NumberFormatException er){  
        JOptionPane.showMessageDialog( msgframe, "Card ID should  
        return;  
    }  
  
    // Sending user a message if he/she enters BalanceAmount in a
```

Figure 39 : Semantic error (Semester 2)

To fix this error I converted the input take from Dcardidtext text field to integer. Now the program will not accept Card Id in any other format than integer.

```
else {  
    // Sending user a message if he/she enters CardID in any form oth  
    try{  
        int Dcardid1data = Integer.parseInt(Dcardidtext.getText());  
    }  
    catch(NumberFormatException er){  
        JOptionPane.showMessageDialog( msgframe, "Card ID should be :  
        return;  
    }  
}
```

Figure 40 : Semantic error correction (Semester 2)

c)Error 3 (Logical Error)

Here while checking if there is enough balance to withdraw instead of writing Balance amount is greater than or equal to withdrawal amount, I wrote Balance amount is smaller than or equal to withdrawal. The program didn't show any error while compiling but the result was different than expected. If I try to withdraw more money than the balance amount, the program should show an error, which obviously didn't happen because I made a mistake in my code.

```
// Checking if there is enough balance
if(Ddetails1.getBalanceAmount() <= Dwithdrawalamountdata){

    //Withdrawing the money
    Ddetails1.Withdraw(Dwithdrawalamountdata,withdrawaltime, Dpin
    String Data = "CardID : " + Dcardid2data + " " + "\nPin Number
                "\nWithdraw Amount :" + Dwithdrawalamountdata + " " +

    Ddetails1.setbalanceamount(Dbalanceamountdata - Dwithdrawalamo
    JOptionPane.showMessageDialog( msgframe, "Sucessfully withdraw
```

Figure 41 : Logical error (Semester 2)

I fixed this logical error by changing the greater than or equal to sign to smaller than or equal to sign. Now the program works perfectly as it should.

```
// Checking if there is enough balance
if(Ddetails1.getBalanceAmount() >= Dwithdrawalamountdata){

    //Withdrawing the money
    Ddetails1.Withdraw(Dwithdrawalamountdata,withdrawaltime, Dpin
    String Data = "CardID : " + Dcardid2data + " " + "\nPin Numbe
                "\nWithdraw Amount :" + Dwithdrawalamountdata + " " +

    Ddetails1.setbalanceamount(Dbalanceamountdata - Dwithdrawalam
```

Figure 42 : Logical error correction (Semester 2)

Conclusion

This course work was a little more interesting than the last one. We had to work on our previous project and add a graphic user interface (GUI) that takes input from the user. Working with GUI was more fun than I expected, this time I felt like I did something, I could arrange the labels, text fields and buttons however or wherever I wanted but just because I had more fun this time doesn't mean it was easy. This time the question was even harder than last time, the code was also longer, and it took a lot of time than expected. I had even more difficulties this time, but my teachers, friends and seniors helped me with this project which made things a lot easier. Last time documentation part was a lot more time consuming than the whole coding part but this time I spent majority of my time coding.

I learned a lot of things from this course work. I now know a lot more about java and GUI, I understood how a button works and how we can take the input from the user. I'm now a more skilled programmer than before. I also learned a lot by making this documentation part, I learned about pseudocode, class diagram, referencing, I also learned about different features of MS-Word. The task was hard and even frustrating sometimes but now when I'm about to finish this I'm having this feeling, a feeling of satisfaction I'm going to sleep like a baby today.

References

Anon., n.d. *Introduction to Microsoft Word*. [Online]

Available at: <https://www.geeksforgeeks.org/introduction-to-microsoft-word/>

Anon., n.d. *Java Introduction*. [Online]

Available at: https://www.w3schools.com/java/java_intro.asp

Anon., n.d. *What is BlueJ?*. [Online]

Available at: <https://www.bluej.org/about.html>

Anon., n.d. *What is class diagram?*. [Online]

Available at: <https://www.javatpoint.com/uml-class-diagram>

Anon., n.d. *What is pseudocode*. [Online]

Available at: <https://www.freecodecamp.org/news/what-is-pseudocode-in-programming/>

APPENDIX

a) Bank Card Class Code

```
public class BankCard
{
    //Putting attributes
    private int CardId;
    private String ClientName;
    private String IssuerBank;
    private String BankAccount;
    private int BalanceAmount;

    //Constructor and Parameters

    public BankCard (int CardId, String IssuerBank, String BankAccount, int
BalanceAmount){
        this.CardId = CardId;
        this.IssuerBank = IssuerBank;
        this.BankAccount = BankAccount;
        this.BalanceAmount = BalanceAmount;
        this.ClientName = "";
    }

    // accessor method of each attributes
    public int getCardId(){
        return CardId;
    }
}
```

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

```
public String getIssuerBank(){  
    return IssuerBank;  
}
```

```
public String getBankAccount(){  
    return BankAccount;  
}
```

```
public int getBalanceAmount(){  
    return BalanceAmount;  
}
```

```
//Method to set client name and balance ammonut  
public void setclientname(String newName){  
    this.ClientName = newName;  
}
```

```
public void setbalanceamount(int newAmount){  
    this.BalanceAmount = newAmount;  
}
```

```
// display method to output all the attributes and a suitable message if the client name  
is not assigned
```

```
public void display(){
```



```
if(ClientName.equals("")){  
    System.out.println("Client name is not given");  
}  
else{  
    System.out.println("Card Id = " + CardId);  
    System.out.println("Client Name = " + ClientName);  
    System.out.println("Issuer Bank = " + IssuerBank);  
    System.out.println("Bank Account = " + BankAccount);  
    System.out.println("Balance Amount = " + BalanceAmount);  
}  
}  
}
```

b) Debit Card Class Code

```
public class DebitCard extends BankCard
{
    //Putting attributes
    private int PINnumber;
    private int WithdrawalAmount;
    private String DateOfWithdrawal;
    private boolean HasWithdrawn;

    //Constructor and Parameters
    public DebitCard(int BalanceAmount, int CardId, String BankAccount, String
IssuerBank,String ClientName, int PINnumber){
        super( CardId, IssuerBank, BankAccount, BalanceAmount);
        setclientname(ClientName);
        this.PINnumber = PINnumber;
        HasWithdrawn = false;
    }

    // acessor method of each attributes
    public int getPINnumber(){
        return PINnumber;
    }

    public int getWithdrawalAmount(){
        return WithdrawalAmount;
    }

    public String getDateOfWithdrawal(){
```

```

        return DateOfWithdrawal;
    }

    public boolean getHasWithdrawn(){
        return HasWithdrawn;
    }

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

    //Withdraw method that deducts the money directly from the client account
    public void Withdraw(int WithdrawalAmount, String DateOfWithdrawal, int
PINnumber){
        if(PINnumber == this.PINnumber){
            if(getBalanceAmount() >= WithdrawalAmount){
                setbalanceamount(getBalanceAmount() - WithdrawalAmount);
                this.WithdrawalAmount = WithdrawalAmount;
                this.DateOfWithdrawal = DateOfWithdrawal;
                this.HasWithdrawn = true;
                System.out.println("Withdrawal Sucessful");
            }
            else if(getBalanceAmount() < WithdrawalAmount){
                System.out.println("Insufficient Balance");
            }
            else if(PINnumber != this.PINnumber){
                System.out.println("Invalid PIN number");
            }
        }
    }
}

```

```
}

// method that displays the details of Debit card
public void display(){
    super.display();
    System.out.println("PIN = " + PINnumber);
    if (HasWithdrawn = true){
        System.out.println("Withdrawal Amount = " + WithdrawalAmount);
        System.out.println("Date of Withdrawal = " + DateOfWithdrawal);
    }
    else{
        System.out.println("No transaction yet");
        System.out.println("Balance Amount = " + super.getBalanceAmount());
    }
}
}
```

c) Credit Card Class Code

```
public class CreditCard extends BankCard
{
    //Putting attributes
    private int CVCnumber;
    private double CreditLimit;
    private double InterestRate;
    private String ExpirationDate;
    private int GracePeriod;
    private boolean isGranted;

    //Constructor
    public CreditCard (int CardId, String ClientName, String IssuerBank, String
    BankAccount, int BalanceAmount, int CVCnumber, double InterestRate,String
    ExpirationDate){
        super(CardId, IssuerBank, BankAccount, BalanceAmount);
        setclientname(ClientName);
        this.CVCnumber = CVCnumber;
        this.InterestRate = InterestRate;
        this.ExpirationDate = ExpirationDate;
        this.isGranted = false;
    }

    //acessor method of each attributes
    public int getCVCnumber(){
        return CVCnumber;
    }
}
```

```
public double getCreditLimit(){  
    return CreditLimit;  
}
```

```
public double getInterestRate(){  
    return InterestRate;  
}
```

```
public String getExpirationDate(){  
    return ExpirationDate;  
}
```

```
public int getGracePeriod(){  
    return GracePeriod;  
}
```

```
public boolean getisGranted(){  
    return isGranted;  
}
```

```
// method that sets credit limit
```

```
public void setCreditLimit(double CreditLimit, int GracePeriod){  
    if (CreditLimit <= 2.5 * super.getBalanceAmount()){  
        this.CreditLimit = CreditLimit;  
        this.GracePeriod = GracePeriod;  
        this.isGranted = true;  
    }  
    else{
```

```

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

//method that cancels the credit card when certain conditions are met
public void cancelCreditCard(){
    this.CVCnumber = 0;
    this.CreditLimit = 0;
    this.GracePeriod = 0;
    this.isGranted = false;
}

//method that displays the details of the Credit card only when isGranted is true
public void display(){
    if(isGranted = true){
        super.display();
        System.out.println("Credit Limit = " + CreditLimit);
        System.out.println("Grace Period = " + GracePeriod);
    }
    else{
        System.out.println("Credit not Granted");
    }
}
}

```

d) BankGUI Class Code

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

/**
 * Write a description of class GUI here.
 *
 * @author (LondonmetID 22067674 Atal Gyawali)
 * @version (a version number or a date)
 */
public class BankGUI implements ActionListener{
    private JButton adddebit, addcredit, withdrawdebit, cancelcredit, setcreditlimit,
debitclear, creditclear, debitdisplay, creditdisplay;
    private JTextField Dcardidtext, Dclientnametext, Dissuerbanktext,
Dbankaccounttext, Dbalanceamounttext, Dpinnumbertext, Dcardid2text,
Dpinnumber2text, withdrawalamounttext, Ccardidtext, Cclientnametext,
Cissuerbanktext, Cbankaccounttext, Cbalanceamounttext, cvctext,
interestratetext, Ccardid2text, newcreditlimittext, newgraceperiodtext;
    private JComboBox Dyearcombo, Dmonthcombo, Ddaycombo, Cyearcombo,
Cmonthcombo, Cdaycombo;
    private ArrayList<BankCard> carddetails;
    public BankGUI(){

        carddetails = new ArrayList();
        //1. Creating JFrame
        JFrame frame = new JFrame();

        // 1.1 Creating Frame Components

        //1.1.1 Creating Buttons

        adddebit = new JButton("Add Debit Card");
        addcredit = new JButton("Add Credit Card");
        withdrawdebit = new JButton("Withdraw From Debit Card");
        cancelcredit = new JButton("Cancel Credit Card");
        setcreditlimit = new JButton("Set Credit Card");
        debitclear = new JButton("Clear");
        creditclear = new JButton("Clear");
        debitdisplay = new JButton("Display");
```



```
creditdisplay = new JButton("Display");
```

//1.1.2 Creating Labels

```
    //(FOR DEBIT)//  
    JLabel debit = new JLabel("DEBIT");  
    JLabel Dcardid = new JLabel("Card ID:");  
    JLabel Dclientname = new JLabel("Client Name:");  
    JLabel Dissuerbank = new JLabel("Issuer Bank:");  
    JLabel Dbankaccount = new JLabel("Bank Account:");  
    JLabel Dbalanceamount = new JLabel("Balance Amount:");  
    JLabel Dpinnumber = new JLabel("PIN Number:");  
    JLabel Dcardid2 = new JLabel("Card ID:");  
    JLabel Dpinnumber2 = new JLabel("PIN Number:");  
    JLabel withdrawalamount = new JLabel("Withdrawal Amount:");  
    JLabel date = new JLabel("Date of Withdrawal:");
```

```
    //(FOR CREDIT)//  
    JLabel credit = new JLabel("CREDIT");  
    JLabel Ccardid = new JLabel("Card ID:");  
    JLabel Cclientname = new JLabel("Client Name:");  
    JLabel Cissuerbank = new JLabel("Issuer Bank:");  
    JLabel Cbankaccount = new JLabel("Bank Account:");  
    JLabel Cbalanceamount = new JLabel("Balance Amount:");  
    JLabel cvc = new JLabel("CVC Number:");  
    JLabel intererate = new JLabel("Interest Rate:");  
    JLabel expirationdate = new JLabel("Expiration Date:");  
    JLabel Ccardid2 = new JLabel("Card ID:");  
    JLabel newcreditlimit = new JLabel("New Credit Limit:");  
    JLabel newgraceperiod = new JLabel("New Grace Period:");
```

//1.1.3 Creating Text Fields

```
    //(FOR DEBIT)//  
    Dcardidtext = new JTextField();  
    Dclientnametext = new JTextField();  
    Dissuerbanktext = new JTextField();  
    Dbankaccounttext = new JTextField();  
    Dbalanceamounttext = new JTextField();  
    Dpinnumbertext = new JTextField();  
    Dcardid2text = new JTextField();  
    Dpinnumber2text = new JTextField();
```

```

withdrawalamounttext = new JTextField();

//(FOR CREDIT)//
Ccardidtext = new JTextField();
Cclientnametext = new JTextField();
Cissuerbanktext = new JTextField();
Cbankaccounttext = new JTextField();
Cbalanceamounttext = new JTextField();
cvctext = new JTextField();
interestratetext = new JTextField();
Ccardid2text = new JTextField();
newcreditlimittext = new JTextField();
newgraceperiodtext = new JTextField();

//1.1.4 Creating Combo Box
//(FOR DEBIT)
Dyearcombo = new JComboBox();
for(int dy = 2023; dy >= 1990; dy--){
    Dyearcombo.addItem(dy);
}

String[] month =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
Dmonthcombo = new JComboBox(month);

Ddaycombo = new JComboBox();
for(int dd = 1; dd <= 31; dd++){
    Ddaycombo.addItem(dd);
}

//(FOR CREDIT)
Cyearcombo = new JComboBox();
for(int cy = 2023; cy >= 1990; cy--){
    Cyearcombo.addItem(cy);
}

Cmonthcombo = new JComboBox(month);

Cdaycombo = new JComboBox();
for(int cd = 1; cd <= 31; cd++){
    Cdaycombo.addItem(cd);
}

```

// 1.2 Setting Components Bounds

//1.2.1 Setting (Button) Bounds

```
adddebit.setBounds(1019,56,174,76);
addcredit.setBounds(999,502,214,39);
withdrawdebit.setBounds(936,227,191,64);
cancelcredit.setBounds(906,632,190,62);
setcreditlimit.setBounds(906,568,201,56);
debitclear.setBounds(1148,226,105,65);
creditclear.setBounds(1123,568,114,56);
debitdisplay.setBounds(1014,303,190,45);
creditdisplay.setBounds(1107,632,130,62);
```

//1.2.2 Setting (Label) Bounds

```
//(FOR DEBIT)//
debit.setBounds(575,0,130,67);
Dcardid.setBounds(34,78,126,31);
Dclientname.setBounds(34,120,140,31);
Dissuerbank.setBounds(35,162,139,31);
Dbankaccount.setBounds(454,78,171,31);
Dbalanceamount.setBounds(454,123,186,31);
Dpinnumber.setBounds(454,168,169,31);
Dcardid2.setBounds(31,233,120,31);
Dpinnumber2.setBounds(31,283,142,31);
withdrawalamount.setBounds(442,229,215,28);
date.setBounds(449,283,208,31);
```

// (FONT) //

```
debit.setFont(new Font("Arial", Font.PLAIN, 36));
Dcardid.setFont(new Font("Arial", Font.PLAIN, 22));
Dclientname.setFont(new Font("Arial", Font.PLAIN, 22));
Dissuerbank.setFont(new Font("Arial", Font.PLAIN, 22));
Dbankaccount.setFont(new Font("Arial", Font.PLAIN, 22));
Dbalanceamount.setFont(new Font("Arial", Font.PLAIN, 22));
Dpinnumber.setFont(new Font("Arial", Font.PLAIN, 22));
Dcardid2.setFont(new Font("Arial", Font.PLAIN, 22));
Dpinnumber2.setFont(new Font("Arial", Font.PLAIN, 22));
withdrawalamount.setFont(new Font("Arial", Font.PLAIN, 22));
date.setFont(new Font("Arial", Font.PLAIN, 22));
```

//(FOR CREDIT)//

```

credit.setBounds(581,353,150,58);
Ccardid.setBounds(24,424,116,31);
Cclientname.setBounds(24,467,139,31);
Cissuerbank.setBounds(24,510,139,31);
Cbankaccount.setBounds(393,423,169,31);
Cbalanceamount.setBounds(393,465,186,31);
cvc.setBounds(393,509,169,31);
interestrate.setBounds(833,422,169,31);
expirationdate.setBounds(833,466,169,31);
Ccardid2.setBounds(32,584,109,31);
newcreditlimit.setBounds(410,573,191,31);
newgraceperiod.setBounds(410,624,204,31);

```

```

//(FONT)//
    credit.setFont(new Font("Arial", Font.PLAIN, 36));
    Ccardid.setFont(new Font("Arial", Font.PLAIN, 22));
    Cclientname.setFont(new Font("Arial", Font.PLAIN, 22));
    Cissuerbank.setFont(new Font("Arial", Font.PLAIN, 22));
    Cbankaccount.setFont(new Font("Arial", Font.PLAIN, 22));
    Cbalanceamount.setFont(new Font("Arial", Font.PLAIN, 22));
    cvc.setFont(new Font("Arial", Font.PLAIN, 22));
    interestrate.setFont(new Font("Arial", Font.PLAIN, 22));
    expirationdate.setFont(new Font("Arial", Font.PLAIN, 22));
    Ccardid2.setFont(new Font("Arial", Font.PLAIN, 22));
    newcreditlimit.setFont(new Font("Arial", Font.PLAIN, 22));
    newgraceperiod.setFont(new Font("Arial", Font.PLAIN, 22));

```

//1.2.3 Setting (Text Field) Bounds

```

//(FOR DEBIT)//
Dcardidtext.setBounds(181,78,242,31);
Dclientnametext.setBounds(181,120,242,31);
Dissuerbanktext.setBounds(181,162,242,31);
Dbankaccounttext.setBounds(648,78,242,31);
Dbalanceamounttext.setBounds(648,123,242,31);
Dpinnumbertext.setBounds(648,170,242,31);
Dcardid2text.setBounds(173,233,250,31);
Dpinnumber2text.setBounds(173,285,250,31);
withdrawalamounttext.setBounds(664,225,250,31);

```

```

// (FONT) //
    Dcardidtext.setFont(new Font("Arial", Font.PLAIN, 22));
    Dclientnametext.setFont(new Font("Arial", Font.PLAIN, 22));

```

```

Dissuerbanktext.setFont(new Font("Arial", Font.PLAIN, 22));
Dbankaccounttext.setFont(new Font("Arial", Font.PLAIN, 22));
Dbalanceamounttext.setFont(new Font("Arial", Font.PLAIN,
22));

Dpinnumbertext.setFont(new Font("Arial", Font.PLAIN, 22));
Dcardid2text.setFont(new Font("Arial", Font.PLAIN, 22));
Dpinnumber2text.setFont(new Font("Arial", Font.PLAIN, 22));
withdrawalamounttext.setFont(new Font("Arial", Font.PLAIN,
22));

//(FOR CREDIT)//
Ccardidtext.setBounds(167,423,210,31);
Cclientnametext.setBounds(167,466,210,31);
Cissuerbanktext.setBounds(167,509,210,31);
Cbankaccounttext.setBounds(595,417,224,31);
Cbalanceamounttext.setBounds(595,465,224,31);
cvctext.setBounds(595,513,224,31);
interestratetext.setBounds(1016,418,224,31);
Ccardid2text.setBounds(151,581,227,31);
newcreditlimittext.setBounds(625,572,212,31);
newgraceperiodtext.setBounds(625,624,212,31);

//(FONT)//
Ccardidtext.setFont(new Font("Arial", Font.PLAIN, 22));
Cclientnametext.setFont(new Font("Arial", Font.PLAIN, 22));
Cissuerbanktext.setFont(new Font("Arial", Font.PLAIN, 22));
Cbankaccounttext.setFont(new Font("Arial", Font.PLAIN, 22));
Cbalanceamounttext.setFont(new Font("Arial", Font.PLAIN,
22));

cvctext.setFont(new Font("Arial", Font.PLAIN, 22));
interestratetext.setFont(new Font("Arial", Font.PLAIN, 22));
Ccardid2text.setFont(new Font("Arial", Font.PLAIN, 22));
newcreditlimittext.setFont(new Font("Arial", Font.PLAIN, 22));
newgraceperiodtext.setFont(new Font("Arial", Font.PLAIN, 22));

//1.2.4 Setting (Combo Box) Bounds
//(FOR DEBIT)
Dyearcombo.setBounds(664,283,70,31);
Dmonthcombo.setBounds(752,283,70,31);
Ddaycombo.setBounds(844,283,70,31);

//(FOR CREDIT)
Cyearcombo.setBounds(1007,462,69,31);

```

```
Cmonthcombo.setBounds(1094,462,69,31);
Cdaycombo.setBounds(1185,462,69,31);
```

// 1.3 Adding Frame Components And Action Listener

```
//Adding Button Action Listener
adddebit.addActionListener(this);
addcredit.addActionListener(this);
withdrawdebit.addActionListener(this);
cancelcredit.addActionListener(this);
setcreditlimit.addActionListener(this);
debitclear.addActionListener(this);
creditclear.addActionListener(this);
debitdisplay.addActionListener(this);
creditdisplay.addActionListener(this);
```

//1.3.1 Adding (Buttons)

```
frame.add(adddebit);
frame.add(addcredit);
frame.add(withdrawdebit);
frame.add(cancelcredit);
frame.add(setcreditlimit);
frame.add(debitclear);
frame.add(creditclear);
frame.add(debitdisplay);
frame.add(creditdisplay);
```

//1.3.2 Adding (Labels)

```
//(FOR DEBIT)//
frame.add(debit);
frame.add(Dcardid);
frame.add(Dclientname);
frame.add(Dissuerbank);
frame.add(Dbankaccount);
frame.add(Dbalanceamount);
frame.add(Dpinnumber);
frame.add(Dcardid2);
frame.add(Dpinnumber2);
frame.add(withdrawalamount);
frame.add(date);
```

```

        //(FOR CREDIT)//
        frame.add(credit);
        frame.add(Ccardid);
        frame.add(Cclientname);
        frame.add(Cissuerbank);
        frame.add(Cbankaccount);
        frame.add(Cbalanceamount);
        frame.add(cvc);
        frame.add(interestrate);
        frame.add(expirationdate);
        frame.add(Ccardid2);
        frame.add(newcreditlimit);
        frame.add(newgraceperiod);

// Adding (Text Field)

        // Adding TextField Action Listener
        //(Debit Listener)//
        Dcardidtext.addActionListener(this);
        Dclientnametext.addActionListener(this);
        Dissuerbanktext.addActionListener(this);
        Dbankaccounttext.addActionListener(this);
        Dbalanceamounttext.addActionListener(this);
        Dpinnumbertext.addActionListener(this);
        Dcardid2text.addActionListener(this);
        Dpinnumber2text.addActionListener(this);
        withdrawalamounttext.addActionListener(this);

        //(FOR DEBIT)//
        frame.add(Dcardidtext);
        frame.add(Dclientnametext);
        frame.add(Dissuerbanktext);
        frame.add(Dbankaccounttext);
        frame.add(Dbalanceamounttext);
        frame.add(Dpinnumbertext);
        frame.add(Dcardid2text);
        frame.add(Dpinnumber2text);
        frame.add(withdrawalamounttext);

        // Adding TextField Action Listener
        //(Credit Listener)//

        Ccardidtext.addActionListener(this);

```

```
Cclientnametext.addActionListener(this);
Cissuerbanktext.addActionListener(this);
Cbankaccounttext.addActionListener(this);
Cbalanceamounttext.addActionListener(this);
cvctext.addActionListener(this);
interestratetext.addActionListener(this);
Ccardid2text.addActionListener(this);
newcreditlimittext.addActionListener(this);
newgraceperiodtext.addActionListener(this);
```

```
//(FOR CREDIT)
frame.add(Ccardidtext);
frame.add(Cclientnametext);
frame.add(Cissuerbanktext);
frame.add(Cbankaccounttext);
frame.add(Cbalanceamounttext);
frame.add(cvctext);
frame.add(interestratetext);
frame.add(Ccardid2text);
frame.add(newcreditlimittext);
frame.add(newgraceperiodtext);
```

```
// Adding (Combo Box)
```

```
// Adding combo box action listener
// ( Debit combo)
Dyearcombo.addActionListener(this);
Dmonthcombo.addActionListener(this);
Ddaycombo.addActionListener(this);
```

```
//(FOR DEBIT)
frame.add(Dyearcombo);
frame.add(Dmonthcombo);
frame.add(Ddaycombo);
```

```
// Adding combo box action listener
//( Credit Combo)
Cyearcombo.addActionListener(this);
Cmonthcombo.addActionListener(this);
Cdaycombo.addActionListener(this);
```

```
//(FOR CREDIT)
```



```

        frame.add(Cyearcombo);
        frame.add(Cmonthcombo);
        frame.add(Cdaycombo);

//2. Setting Frame Size
frame.setSize(1280,750);

//3. Setting Layout
frame.setLayout(null);

//4. Setting Default operation
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//5. Setting Visibility
frame.setVisible(true);

//6. Setting Resizability
frame.setResizable(false);
}

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

// Giving Functionality to Buttons
JFrame msgframe = new JFrame();

public void actionPerformed(ActionEvent e){

    // ADD Debit Button
    if(e.getSource()== adddebit){
        if(Dcardidtext.getText().isEmpty() | Dclientnametext.getText().isEmpty() |
Dissuerbanktext.getText().isEmpty() |
        Dbankaccounttext.getText().isEmpty() |
Dbalanceamounttext.getText().isEmpty() | Dpinnumbertext.getText().isEmpty()){
            JOptionPane.showMessageDialog( msgframe, "Please Input every
field");
        }
        else {
            // Sending user a message if he/she enters CardID in any form other
than integer
            try{

```

```

        int Dcardid1data = Integer.parseInt(Dcardidtext.getText());
    }
    catch(NumberFormatException er){
        JOptionPane.showMessageDialog( msgframe, "Card ID should be in
number Format");
        return;
    }

    // Sending user a message if he/she enters BalanceAmount in any
form other than Integer
    try{
        int Dbalanceamountdata =
Integer.parseInt(Dbalanceamounttext.getText());
    }
    catch(NumberFormatException er){
        JOptionPane.showMessageDialog( msgframe, "Balance Amount
should be in number Format");
        return;
    }

    // Sending user a message if he/she enters PIN Number in any form
other than integer
    try{
        int Dpinnumber1data = Integer.parseInt(Dpinnumbertext.getText());
    }
    catch(NumberFormatException er){
        JOptionPane.showMessageDialog( msgframe, "PIN Number should
be in number Format");
        return;
    }
    // Storing Inputs in Variable
    int Dcardid1data = Integer.parseInt(Dcardidtext.getText());
    String Dissuerbankdata = Dissuerbanktext.getText();
    String Dbankaccountdata = Dbankaccounttext.getText();
    int Dbalanceamountdata =
Integer.parseInt(Dbalanceamounttext.getText());
    int Dpinnumber1data = Integer.parseInt(Dpinnumbertext.getText());
    String Dclientnamedata = Dclientnametext.getText();
    int repeat = 0;

    if(carddetails.size() == 0){
        // Storing the data in Array list

```

```

        DebitCard Ddetails1 = new DebitCard
(Dbalanceamountdata,Dcardid1data,Dbankaccountdata,Dissuerbankdata,Dclientn
amedata,
        Dpinnumber1data);
        carddetails.add(Ddetails1);
        JOptionPane.showMessageDialog( msgframe, "Debit Card Added");
    }
    else{
        // Checking if the debit card already exists
        //for BankCard variable in carddetails array
        for(BankCard details : carddetails){
            if (details instanceof DebitCard){
                DebitCard Ddetails2 = (DebitCard) details;
                if (Ddetails2.getCardId() == Dcardid1data){
                    repeat = repeat + 1;
                }
            }
        }
        // If the debit card doesn't exist then adding the debit card
        if (repeat == 0){
            DebitCard Ddetails3 = new
DebitCard(Dbalanceamountdata,Dcardid1data,Dbankaccountdata,Dissuerbankdat
a,Dclientnamedata,
            Dpinnumber1data);
            carddetails.add(Ddetails3);
            JOptionPane.showMessageDialog( msgframe, "Debit Card
Added");
        }
        // sending message to the user if the Debit Card already exists
        else{
            JOptionPane.showMessageDialog( msgframe, "Debit Card
Already Exists");
        }
    }
}

// Withdraw From Debit Card Button
if(e.getSource() == withdrawdebit){
    String withdrawalttime = Dyearcombo.getSelectedItem() + "/" +
Dmonthcombo.getSelectedItem()+ "/" + Ddaycombo.getSelectedItem();
    if(Dcardid2text.getText().isEmpty() | Dpinnumber2text.getText().isEmpty()
| withdrawalamounttext.getText().isEmpty()){

```

```

        JOptionPane.showMessageDialog( msgframe, "Please Input every
field");
    }
    else{
        // Sending user a message if he/she enters Card ID in any form other
than integer
        try{
            int Dcardid2data = Integer.parseInt(Dcardid2text.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "Card ID should be in
number Format");
            return;
        }

        // Sending user a message if he/she enters PIN Number in any form
other than integer
        try{
            int Dpinnumeber2data =
Integer.parseInt(Dpinnumber2text.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "PIN Number should
be in number Format");
            return;
        }

        // Sending user a message if he/she enters Withdrawal Amount in any
form other than integer
        try{
            int Dwithdrawalamountdata =
Integer.parseInt(withdrawalamounttext.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "Withdrawal Amount
should be in number Format");
            return;
        }

        // Storing Inputs in Variables
        int Dcardid2data = Integer.parseInt(Dcardid2text.getText());
        int Dpinnumber2data = Integer.parseInt(Dpinnumber2text.getText());

```

```

        int Dwithdrawalamountdata =
Integer.parseInt(withdrawalamounttext.getText());
        boolean cardExists = false;
        int Dbalanceamountdata =
Integer.parseInt(Dbalanceamounttext.getText());

        // Iterating over debit card
        for ( BankCard details : carddetails){
            if(details instanceof DebitCard){
                DebitCard Ddetails1 = (DebitCard) details;

                // Checking if the entered ID matches
                if(Ddetails1.getCardId() == Dcardid2data){
                    cardExists = true ;

                    // Checking if the PIN number matches
                    if(Ddetails1.getPINnumber() == Dpinnumber2data){

                        // Checking if there is enough balance
                        if(Ddetails1.getBalanceAmount() >=
Dwithdrawalamountdata){

                            //Withdrawing the money

                            Ddetails1.Withdraw(Dwithdrawalamountdata,withdrawaltime, Dpinnumber2data);
                            String Data = "CardID : " + Dcardid2data + " " + "\nPin
Number : " + Dpinnumber2data +
                                "\nWithdraw Amount : " + Dwithdrawalamountdata + "
" + "\nTime : " + withdrawaltime;

                            Ddetails1.setbalanceamount(Dbalanceamountdata -
Dwithdrawalamountdata);
                            JOptionPane.showMessageDialog( msgframe,
"Succesfully withdrawn from your account" + Data);
                        }
                    }else{
                        JOptionPane.showMessageDialog( msgframe, "Insufficient
Balance in the Account");
                    }
                }
            }
        }
    }
}
else{

```

```

        JOptionPane.showMessageDialog( msgframe, "Wrong Pin
Number");
    }
}
}
}
if (!cardExists) {
    JOptionPane.showMessageDialog( msgframe, "Sorry Your card
doesn't exist in our system");
}
}
}

// Display Button of Debit
if(e.getSource() == debitdisplay){
    // String time = comboboxyear +
    if(carddetails.isEmpty()){
        JOptionPane.showMessageDialog( msgframe, "Please Enter the card
details before displaying them");
    }
    else{
        for(BankCard details : carddetails){
            if ( details instanceof DebitCard){
                DebitCard Ddetails1 = (DebitCard) details;
                String value1 = "Card ID : " + Ddetails1.getCardId()+ "\nClient
Name : " + "\nClient Name: " + Ddetails1.getClientName() +
                "\nIssuer Bank: " + Ddetails1.getIssuerBank() + "\nBalance
Amount: " + Ddetails1.getBalanceAmount() + "\nBank Account: " +
                Ddetails1.getBankAccount() +
                "\nPin Number: " + Ddetails1.getPINnumber() + "\nWithdraw
Amount: " + Ddetails1.getWithdrawalAmount() ;

                String value2 = "Card ID: " + Ddetails1.getCardId() + "\nClient
Name: " + Ddetails1.getClientName() +
                "\nIssuer Bank: " + Ddetails1.getIssuerBank() + "\nBalance
Amount: " + Ddetails1.getBalanceAmount() +
                "\nBank Account: " + Ddetails1.getBankAccount() + "\nPin
Number: " + Ddetails1.getPINnumber() + "\n";

                String message = Ddetails1.getHasWithdrawn() ? "Debit Card.\n"
+ value1 : "Debit Card.\n" + value2 + "\n";

```

```

        // showing the details to the user
        JOptionPane.showMessageDialog( msgframe, message,
"Display", JOptionPane.INFORMATION_MESSAGE);
    }
}

// ADD Credit Button
if(e.getSource() == addcredit){
    if(Ccardidtext.getText().isEmpty() | Cclientnametext.getText().isEmpty() |
Cissuerbanktext.getText().isEmpty() |
    Cbankaccounttext.getText().isEmpty() |
Cbalanceamounttext.getText().isEmpty() | cvctext.getText().isEmpty() |
    interestratetext.getText().isEmpty() ){

        JOptionPane.showMessageDialog( msgframe, "Please Input every
field");
    }
    else{
        // Sending user a message if he/she enters CardID in any form other
than integer
        try{
            int Ccardid1data = Integer.parseInt(Ccardidtext.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "Card ID should be in
number Format");
            return;
        }

        // Sending user a message if he/she enters BalanceAmount in any
form other than Integer
        try{
            int Cbalanceamountdata =
Integer.parseInt(Cbalanceamounttext.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "Balance Amount
should be in number Format");
            return;
        }
    }
}

```

```

        // Sending user a message if he/she enters CVC Number in any form
        other than integer
        try{
            int cvcnumberdata = Integer.parseInt(cvctext.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "CVC Number should
            be in number Format");
            return;
        }

```

```

        // Sending user a message if he/she enters Interest rate in any form
        other than number(double)
        try{
            double interestratedata =
            Double.parseDouble(interestratedata.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "Interest Rate should
            be in number Format");
            return;
        }

```

```

        // Storing Inputs in Variables
        int Ccardid1data = Integer.parseInt(Ccardidtext.getText());
        String Cclientnamedata = Cclientnametext.getText();
        String Cissuerbankdata = Cissuerbanktext.getText();
        String Cbankaccountdata = Cbankaccounttext.getText();
        int Cbalanceamountdata =
        Integer.parseInt(Cbalanceamounttext.getText());
        int cvcnumberdata = Integer.parseInt(cvctext.getText());
        double interestratedata =
        Double.parseDouble(interestratedata.getText());
        String expirationdatedata = "";
        int repeat = 0;

        if(carddetails.size() == 0){
            // Storing the data in Array list
            CreditCard Cdetails1 = new CreditCard (Ccardid1data,
            Cclientnamedata, Cissuerbankdata,
            Cbankaccountdata, Cbalanceamountdata, cvcnumberdata,
            interestratedata, expirationdatedata);
            carddetails.add(Cdetails1);
        }

```



```

        JOptionPane.showMessageDialog( msgframe, "Credit Card
Added");
    }
    else{
        // Checking if the credit card already exists
        //for BankCard variable in carddetails array
        for(BankCard details : carddetails){
            if (details instanceof CreditCard){
                CreditCard Cdetails2 = (CreditCard) details;
                if (Cdetails2.getCardId() == Ccardid1data){
                    repeat = repeat + 1;
                }
            }
        }
        // If the debit card doesn't exist then adding the debit card
        if (repeat == 0){
            CreditCard Cdetails3 = new CreditCard (Ccardid1data,
Cclientnamedata, Cissuerbankdata,
                Cbankaccountdata, Cbalanceamountdata, cvcnumberdata,
interestratedata, expirationdatedata);
            carddetails.add(Cdetails3);
            JOptionPane.showMessageDialog( msgframe, "Credit Card
Added");
        }
        // sending message to the user if the Debit Card already exists
        else{
            JOptionPane.showMessageDialog( msgframe, "Credit Card
Already Exists");
        }
    }
}

// Credit Display Button
if(e.getSource() == creditdisplay){
    String time = Cyearcombo.getSelectedItem() + "/" +
Cmonthcombo.getSelectedItem() + "/" + Cdaycombo.getSelectedItem();
    if(carddetails.isEmpty()){
        JOptionPane.showMessageDialog( msgframe, "Please Enter the card
details before displaying them");
    }
    else{
        for(BankCard details : carddetails){

```

```

        if ( details instanceof CreditCard){
            CreditCard Cdetails1 = (CreditCard) details;

            String value1 = "Card ID : "+ Cdetails1.getCardId() + "\nClient
Name : " + Cdetails1.getClientName() +
                "\nIssuer Bank : " + Cdetails1.getIssuerBank() + "\nBalance
Amount : " + Cdetails1.getBalanceAmount() +
                "\nBank Account : " + Cdetails1.getBankAccount()
+" \n" + "\nGrace Period : " + Cdetails1.getGracePeriod() +
                "\nCredit Limit : " + Cdetails1.getCreditLimit();

            String value2 = "Card ID : "+Cdetails1.getCardId() + "\nClient
Name : " + Cdetails1.getClientName() +
                "\nIssuer Bank : " + Cdetails1.getIssuerBank() + "\nBalance
Amount : " + Cdetails1.getBalanceAmount() +
                "\nBank Account : " + Cdetails1.getBankAccount() +
"\nCVC Number : " + Cdetails1.getCVCNumber() +
                "\nInterest Rate : " + Cdetails1.getInterestRate() +
                "\nExpiration Date : " + time + "\nIs Granted : " +
Cdetails1.getisGranted() ;

            if(Cdetails1.getisGranted() == true ){
                JOptionPane.showMessageDialog(null,"Credit Card.\n" +
value1 , "Display",
                JOptionPane.INFORMATION_MESSAGE);
            }

            else{
                JOptionPane.showMessageDialog(null,"Credit Card.\n" +
value2 + "\n", "Display",
                JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
}

// Set Credit Limit Button
if(e.getSource() == setcreditlimit){
    if(Ccardid2text.getText().isEmpty() | newcreditlimittext.getText().isEmpty()
| newgraceperiodtext.getText().isEmpty()){
        JOptionPane.showMessageDialog( msgframe, "Please Input every
field");
    }
}

```

```

    }
    else{

        // Sending user a message if he/she enters Card ID in any form other
        than integer
        try{
            int Ccardid2data = Integer.parseInt(Ccardid2text.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "Card ID should be in
            number Format");
            return;
        }

        // Sending user a message if he/she enters Credit Limit in any form other
        than double
        try{
            double newcreditlimitdata =
            Double.parseDouble(newcreditlimittext.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "Credit Limit should be
            in number Format");
            return;
        }

        // Sending user a message if he/she enters Grace period in any form
        other than Integer
        try{
            int newcgraceperioddata =
            Integer.parseInt(newgraceperiodtext.getText());
        }
        catch(NumberFormatException er){
            JOptionPane.showMessageDialog( msgframe, "New Grace Period
            should be in number Format");
            return;
        }

        // Storing Inputs in variables
        int Ccardid2data = Integer.parseInt(Ccardid2text.getText());
        double newcreditlimitdata =
        Double.parseDouble(newcreditlimittext.getText());
    }

```

```

        int newcgraceperioddata =
Integer.parseInt(newgraceperiodtext.getText());
        int repeatsc = 0;

        if(carddetails.size() == 0 ){
            JOptionPane.showMessageDialog( msgframe," Please Enter the
values to set the credit limit "); }

        else{
            for(BankCard Ccard : carddetails){
                if (Ccard instanceof CreditCard){
                    CreditCard Cdetails1 = (CreditCard) Ccard;
                    if(Cdetails1.getCardId() == Ccardid2data ){
                        if(newcreditlimitdata <= 2.5 *
Cdetails1.getBalanceAmount()){

Cdetails1.setCreditLimit(newcreditlimitdata,newcgraceperioddata);
                            String Data = "Card Id : " + Ccardid2data + "Grace Period : "
+ newcgraceperioddata;

                                JOptionPane.showMessageDialog( msgframe," Credit Limit
set");
                                    repeatsc ++ ;
                                }
                            }
                        }
                    }}
                if(repeatsc == 0){
                    JOptionPane.showMessageDialog( msgframe," Sorry your card
doesn't exist in our system");
                }

            }
        }

        // Cancel Credit Card
        if(e.getSource() == cancelcredit){
            if(Ccardidtext.getText().isEmpty() | Cclientnametext.getText().isEmpty() |
Cissuerbanktext.getText().isEmpty() |
                Cbankaccounttext.getText().isEmpty() |
Cbalanceamounttext.getText().isEmpty() | cvctext.getText().isEmpty() |
                    interestratetext.getText().isEmpty() ){

```

```

        JOptionPane.showMessageDialog( msgframe, "Please Input every
field ");
    }
    else
    {
        try{
            int CardID = Integer.parseInt(Ccardidtext.getText());
            //CreditCard
            if(carddetails.size() == 0){
                JOptionPane.showMessageDialog( msgframe, "Sorry your card is
not in our system ");
            }

            else{
                for(BankCard cCard : carddetails){
                    if(cCard instanceof CreditCard){
                        CreditCard Cdetails2 = (CreditCard) cCard;
                        if(Cdetails2.getCardId() == CardID){
                            Cdetails2.cancelCreditCard();
                            JOptionPane.showMessageDialog( msgframe, "Your
credit card has been cancelled ");
                        }
                    }
                }
            }
        }catch(NumberFormatException ex){
            JOptionPane.showMessageDialog( msgframe, "Please Enter the
correct value ");
        }
    }
}

// Debit Clear Button
if(e.getSource() == debitclear){
    Dcardidtext.setText(""); Dclientnametext.setText("");
    Dissuerbanktext.setText(""); Dbankaccounttext.setText("");
    Dbalanceamounttext.setText(""); Dpinnumbertext.setText("");
    Dcardid2text.setText(""); Dpinnumber2text.setText("");
    withdrawalamounttext.setText("");
}

// Credit Clear Button

```

```
if(e.getSource() == creditclear){
    Ccardidtext.setText(""); Cclientnametext.setText(""); ;
    Cissuerbanktext.setText(""); Cbankaccounttext.setText(""); ;
    Cbalanceamounttext.setText(""); cvctext.setText("");
    interestratetext.setText(""); Ccardid2text.setText(""); ;
    newcreditlimittext.setText(""); newgraceperiodtext.setText(""); ;
}

}

}
```