CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Nishan Thapa

London Met ID: 22068741

College ID: NP01CP4A220046

Group – L1C2

Assignment Due Date: Wednesday, May 10, 2023

Assignment Submission Date: Wednesday, May 10, 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.

Table of Contents

Table of Figures

Table of Tables

INTRODUCTION

1.1 Project Introduction

The goal of this project is to use the Java programming language to construct a Graphical User Interface (GUI). The GUI will be created to offer users a platform for interaction with the system and different tasks. As part of the project, several elements will be created, including text fields, combo boxes, and buttons, to let users traverse the system and perform various tasks. The user experience will be prioritized in the design of the GUI, making it intuitive, simple to use, and visually beautiful. The main objective of this project is to create a solid and trustworthy GUI that gives users a quick and easy way to interact with the system.

1.2 Aim of the Project

The Bank GUI project attempts to create a graphical user interface for debit and credit card management. The GUI will offer consumers a convenient and simple environment for doing financial operations. To ensure seamless execution and to handle any unforeseen issues, the project will make use of tabbed pane, button functionality and try-catch exception handling. Users will be able to use the GUI to do tasks such as checking their account balance, withdrawal funds, and modifying their credit limit. The design will be intuitive and user-friendly, allowing clients to

easily explore and use the banking services provided. The purpose of this project is to give consumers with an efficient and safe banking experience via a well-designed GUI that matches their needs and expectations.

1.3 The objective of this Project

This project is carried out to fulfil several objectives which are listed below:

- To create a GUI using java swing and awt.
- To be able to create different Labels, Text Fields, Buttons and Combo Box within the GUI.
- To be able to add functionality to different buttons such as add, display and clear.
- To be able to use try and catch any user exception that may occur.
- To be able to use proper pop-up messages to display when an error occurs.

1.4 Technologies used for the Project

The project was created using a variety of technologies including BlueJ, Microsoft Word, and online portals such as moqups. Moqups was used as a generic wireframe for the Debit and Credit Card GUI. BlueJ was used as a text editor to produce the GUI code, while Microsoft Word was utilized to write the project report.

2 Discussion and Analysis

In general, this section of the report gives a description of the selected Text Editor, where BlueJ has been chosen for coding and compilation. It also includes a description of the technology used to create a wireframe. Moqups was used to develop wireframes for many GUIs of the created GUI.

All of its contents are described below:

2.1 Text Editor

A text editor generally refers to any form of a computer program that allows users to create, update, edit, open, and display plain text files. There are several text editors applications such as Notepad, Sublime Text, VS Code, and many more (University of York, 2023).

Here, BlueJ has been used for editing the code.

Figure 1 BlueJ logo

BlueJ is a Java programming language teaching and learning Integrated Development Environment (IDE). It has a graphical user interface and a range of tools to help students build, test, and debug Java programs. BlueJ was used in our project to create the code for the Bank GUI project. The BlueJ IDE enabled us to design the classes, objects, and methods required to develop the GUI. We were able to test, debug, and modify the code as needed. BlueJ also assisted us in understanding Java object-oriented programming principles such as inheritance and polymorphism. (London, 2023)

2.2 Wireframe Developer

Wireframing is the way of illustrating the way to design a GUI at the structure level. The layout of the page's content and functionality is often done with it. It serves as a foundation for the relationship with GUI templates before any visual design and content are added. It is regarded as the skeletal framework of the project. It develops the blueprints for any website according to

the client's demands. Its main aim is used to provide visual understanding and user-testing to the clients regarding the outlet of websites. (Experienceus, 2023). Various tools like Balsamiq Wireframes, Figma, draw.io, and many more.

Here, moqups and balsamiq wireframe were used for wireframing GUI and class diagram.

Figure 2 moqups logo

Moqups is a web-based tool for developing wireframes and user interface mockups. It has a drag-and-drop interface that allows users to build and customize their designs quickly. Moqups was used in our project to build wireframes for the Bank GUI. Before we started developing, we were able to visualize the layout, design, and functionality of the GUI using the application. It aided us in identifying potential concerns and making essential design adjustments. We were able to design a visually beautiful and user-friendly GUI that fulfilled the demands and expectations of our clients by using Moqups. (S.R.L., 2023)

Figure 3 balsamiq logo

Balsamiq Wireframes are the user interface wireframe design tools used by programmers to sketch their concepts or ideas for designing any websites without writing any code. It provides the blueprint for the structure of the website as per the client's demand. It doesn't include any style, colour, or graphics. It is user-interactive and enables programmers to link their wireframes with the websites. It helps you to pay more attention to structure rather than content details (Kim, 2020)

2.3 Report Developer

The process of preparing a document that delivers information about a specific topic, event, or project is known as report writing. Reports are often written for a specific audience, such as managers, clients, or stakeholders, and are used to update them on the status, progress, or outcomes of a certain project or endeavor. A well-written report should be clear, concise, and organized, with logical information flow that is easy to read and understand. Reports may contain a variety of information, including data analysis, research findings, suggestions, and conclusions. Writing a report needs extensive research, analytical thinking, and great communication abilities.

The purpose of report writing is to communicate difficult information in a clear and succinct manner that the reader can understand.

Figure 4 MS word logo

Word is a commonly used word processing software for writing and modifying reports, papers, and other written content. We used Microsoft Word in our project to create an informative report regarding the Bank GUI project. Word gave us with an easy-to-use interface, several formatting options, and spell-checking capabilities. It enabled us to present the material in a clear and succinct manner, allowing readers to better comprehend the project's objective and scope. (team, 2023)

3 Class Diagram

A class diagram is a graphical representation of the classes and connections that comprise a software application in Java programming. It is used to model ideas in object-oriented programming such as classes, objects, inheritance, associations, and others. A class diagram is a high-level overview of the structure of a system that can be used to plan, document, and convey the system's architecture. A class diagram depicts classes as rectangles with properties and methods listed inside. The relationships between the classes are indicated by arrows between the rectangles and can be classed as associations, aggregations, or generalizations.

3.1 Bank Card (Super Class)

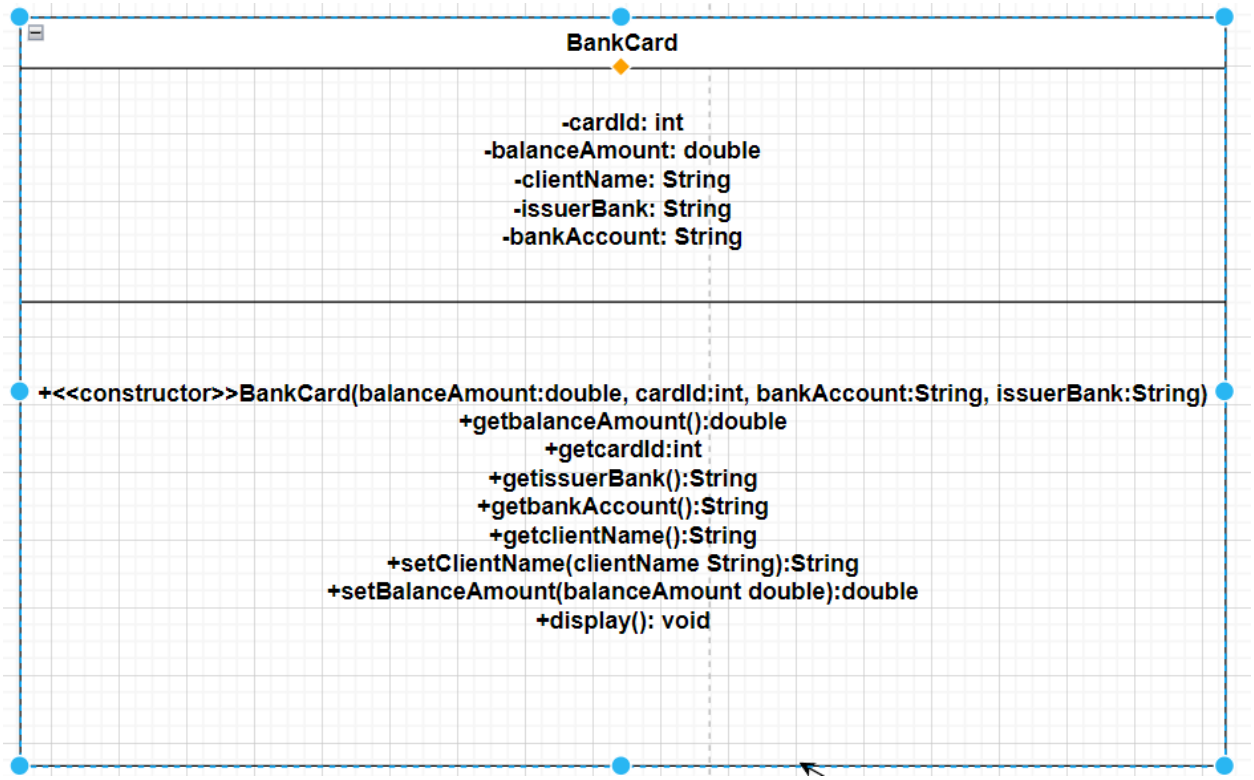

Figure 5 Bank Card class diagram

3.2 Debit Card (Subclass)

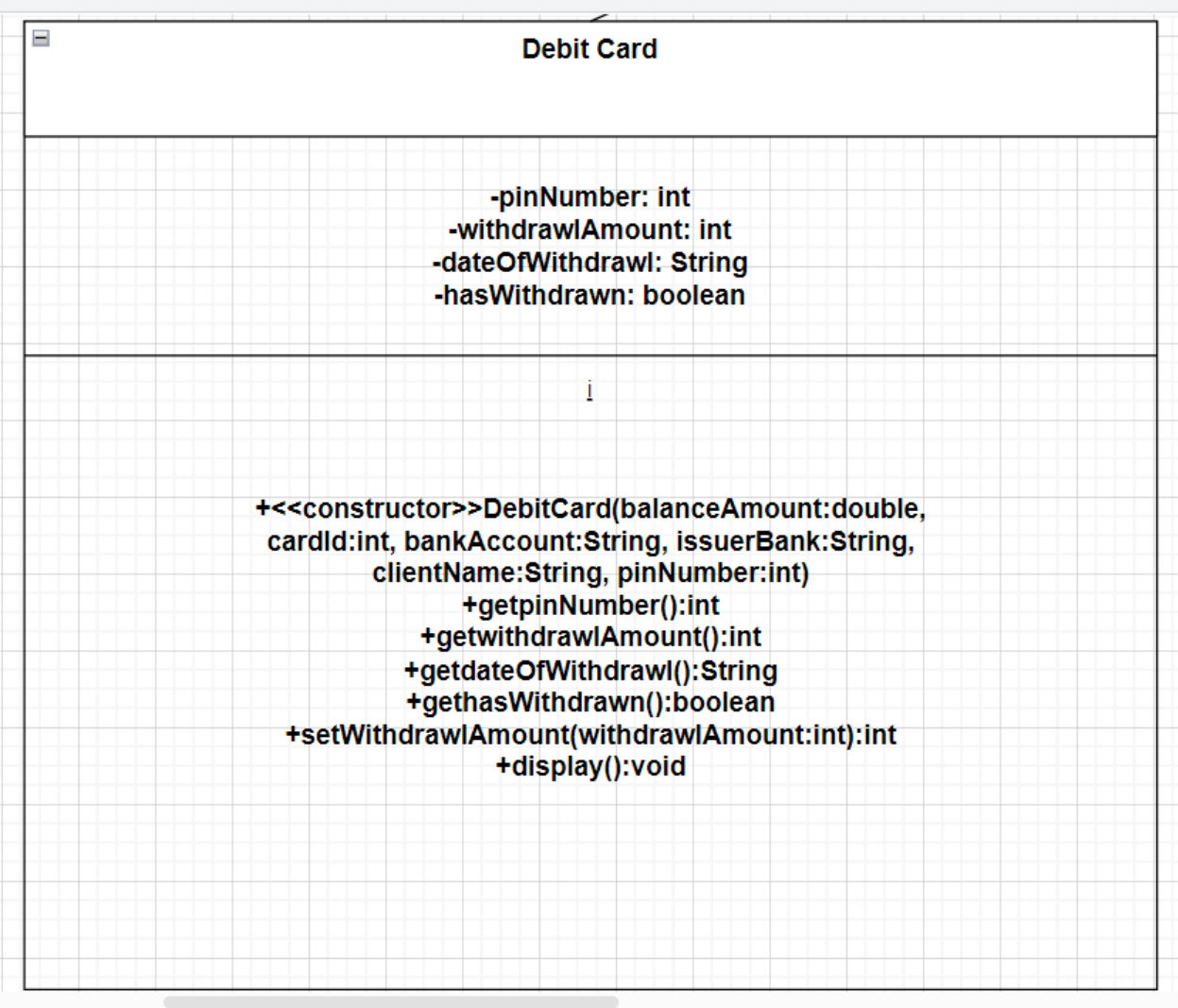

Figure 6 Debit Card class diagram

3.3 Credit Card (Subclass)

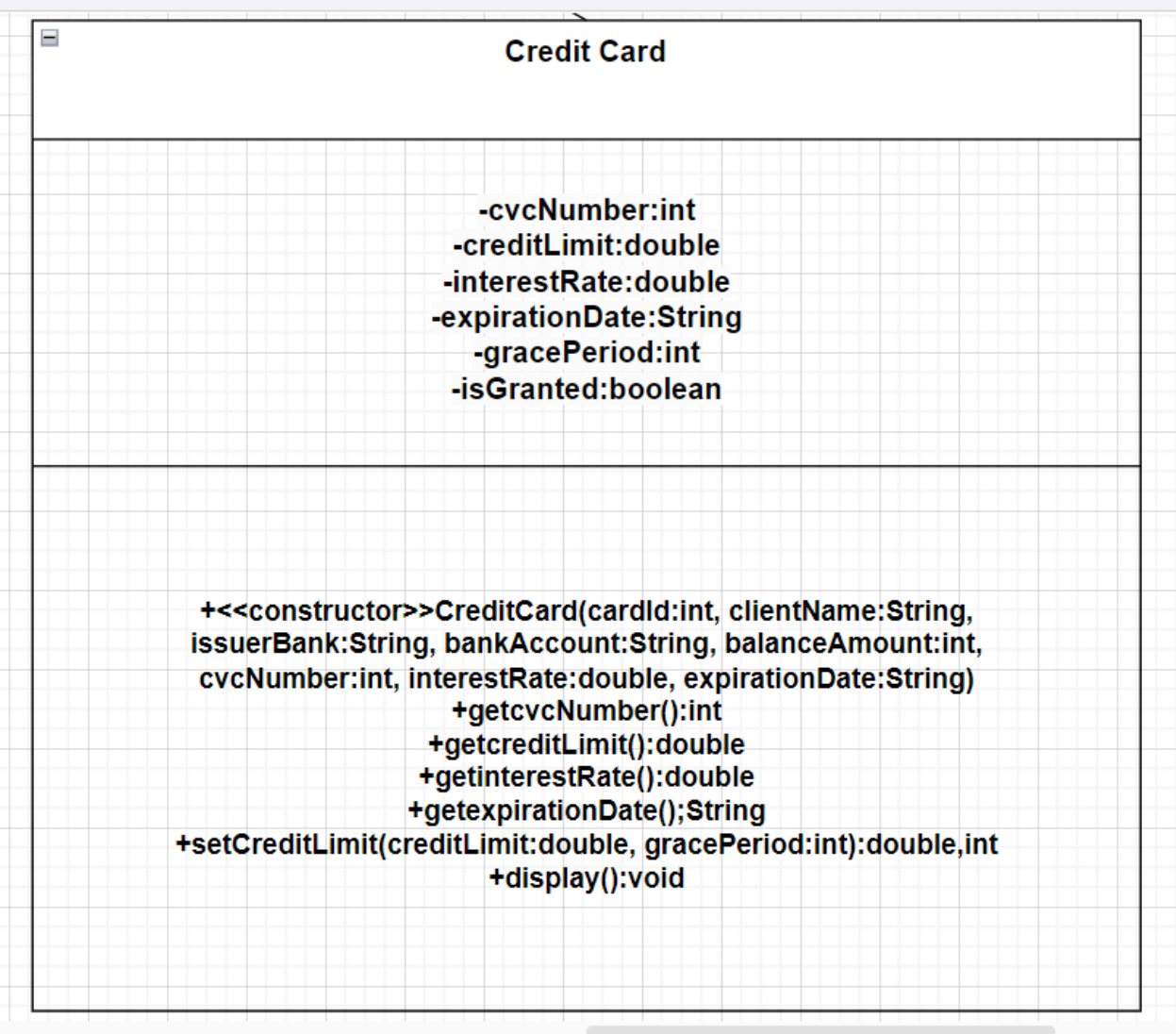

Figure 7 Credit Card class diagram

3.4 Class Diagram representing relation between superclass and subclass

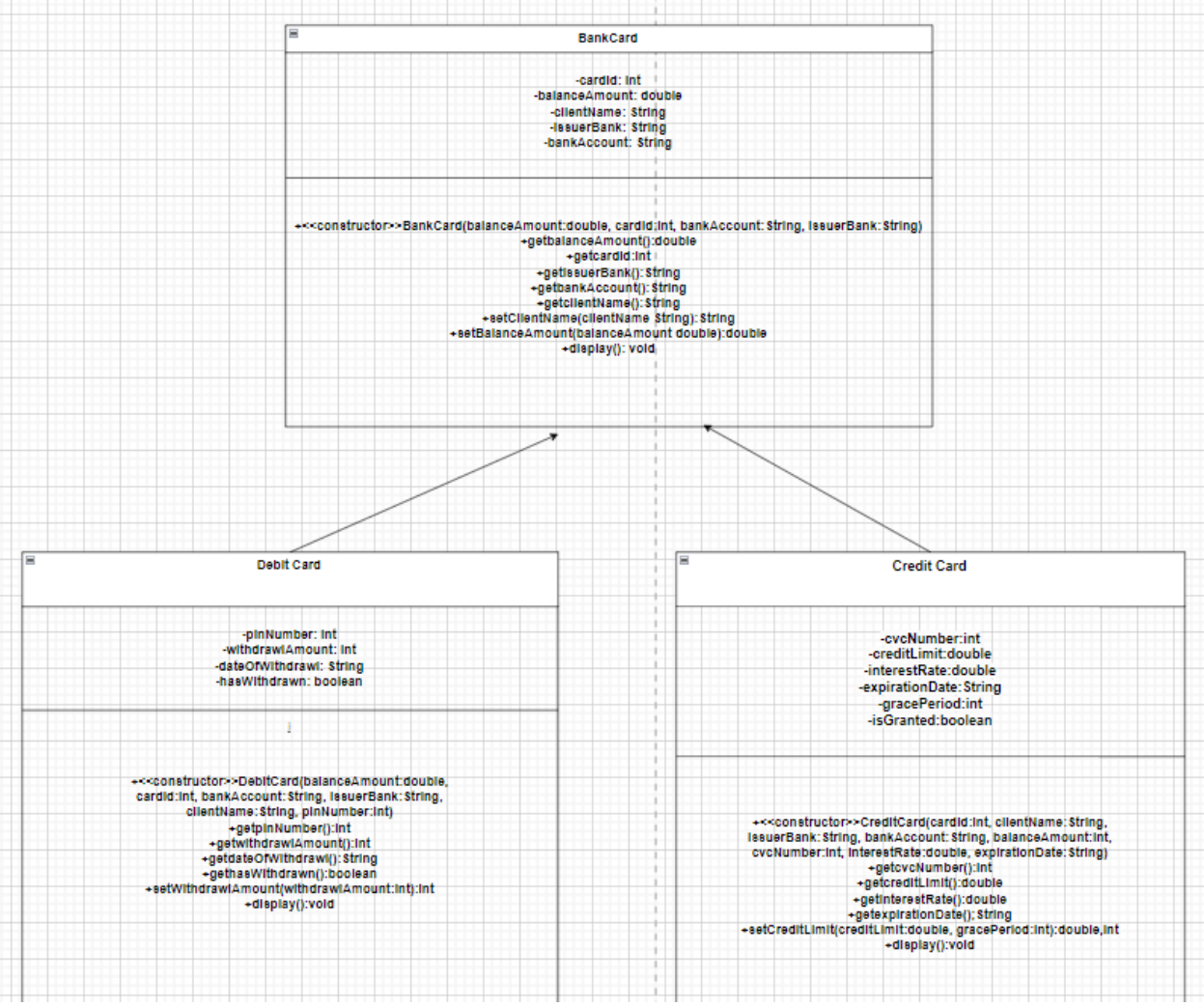

Figure 8 Relationship diagram of Super and subclass

3.5 Bank GUI

Figure 9 Bank GUI class diagram

4 Pseudocode

Java pseudocode is a high-level description of a Java program written in plain English that is used to describe a program's logic without actually utilizing Java syntax. It is a powerful tool in the early phases of software development because it allows developers to concentrate on the problem-solving parts of a program's design rather than being bogged down in the complexities of syntax. Pseudocode is a colloquial language that allows engineers to express complicated ideas in a succinct and understandable manner. Writing Java pseudocode is an important component of the software development process since it helps guarantee that the program is well-structured, efficient, and simple to maintain.

4.1 Bank GUI Pseudocode

IMPORT javax.swing.

IMPORT java.awt.

IMPORT java.awt.event.

IMPORT javax.swing.JPanel

IMPORT java.util.ArrayList

IMPORT javax.swing.JLabel

CALL the BankGUI class that implements ActionListener

DO

DECLARE a frame of JFrame

DECLARE a panel called JTabbedPane

CREATE a debitCard_panel in JPanel

DECLARE a debitCard_clientName, debitCard_issuerBank, debitCard_bankAccount as a String

DECLARE a debitCard_cardID, debitCard_PIN_Number as an int

DECLARE a debitCard_balanceAmount as a double

DECLARE adequate number JLabels for Debit Card as well as Credit Card

DECLARE adequate number JTextFields for Debit Card as well as Credit Card

DECLARE adequate number JButtons for Debit Card as well as Credit Card

DECLARE adequate number JComboBox for Debit Card as well as Credit Card

END DO

CREATE method BankGUI()

DO

INITIALIZE panel

INITIALIZE debitCard_panel

SET debitCard_panel to null

ADD debitCard_panel to panel

INITIALIZE all components of JLabel of Debit Card to debitCard_panel

SET bounds to all components of JLabel of Debit Card to debitCard_panel.

INITIALIZE all components of JTextField of Debit Card to debitCard_panel

SET bounds to all components of JTextField of Debit Card to debitCard_panel.

INITIALIZE all components of JLabel of Withdrawal from Debit card to debitCard_panel

SET bounds to all components of JLabel of Withdrawal from Debit Card to debitCard_panel.

INITIALIZE all components of JTextField of Withdrawal from Debit Card to debitCard_panel

SET bounds to all components of JTextField of Withdrawal from Debit Card to debitCard_panel.

INITIALIZE all components of JComboBox Day, Month and Year to debitCard_panel

SET bounds to all components of JComboBox Day, Month and Year to debitCard_panel.

INITIALIZE all components of JButton of Debit Card and Withdrawal from Debit Card to debitCard_panel

SET bounds to all components of JButton of Debit Card and Withdrawal from Debit Card to debitCard_panel.

ADD all the components of JLabel of Debit Card and Withdrawal from Debit Card to debitCard_panel

ADD all the components of JTextFields of Debit Card and Withdrawal from Debit Card to debitCard_panel

ADD all the components of JComboBox of Debit Card and Withdrawal from Debit Card to debitCard_panel

ADD all the components of JButtons of Debit Card and Withdrawal from Debit Card to debitCard_panel.

INITIALIZE creditCard_panel

SET creditCard_panel to null

ADD creditCard_panel to panel

INITIALIZE all components of JLabel of Credit Card to creditCard_panel

SET bounds to all components of JLabel of Credit Card to creditCard_panel.

INITIALIZE all components of JTextField of Credit Card to creditCard_panel

SET bounds to all components of JTextField of Credit Card to creditCard_panel.

INITIALIZE all components of JLabel of Set Credit card and Cancel credit to creditCard_panel

SET bounds to all components of JLabel of Set Credit card and cancel credit to creditCard_panel.

INITIALIZE all components of JTextField of Set Credit card and cancel credit to creditCard_panel

SET bounds to all components of JTextField of Set Credit card and cancel credit to creditCard_panel.

INITIALIZE all components of JComboBox Day, Month and Year to creditCard_panel

SET bounds to all components of JComboBox Day, Month and Year to creditCard_panel.

INITIALIZE all components of JButton of Credit card, Set Credit card and cancel credit to creditCard_panel

SET bounds to all components of JButton of Credit card, Set Credit card and cancel credit to creditCard_panel.

ADD all the components of JLabel of Credit card, Set Credit card and cancel credit to creditCard_panel

ADD all the components of JTextFields of Credit card, Set Credit card and cancel credit to creditCard_panel

ADD all the components of JComboBox Credit card, Set Credit card and cancel credit to creditCard_panel

ADD all the components of JButtons of Credit card, Set Credit card and cancel credit to creditCard_panel.

ADD frame to the panel

SET the size of the frame

SET the setting of frame to Default on the closing operaton

SET frame visible to true

REGISTER addActionListener to all the components of JButton

END DO

OVERRIDE abstract method actionPerformed (ActionEvent e)

DO

IF (e.getSource() == button for Add Debit Card)

DO

IF debit_clientName, debit_issuerBank , debit_bankAccount, debit_cardID, debit_PIN_Number, debit_balanceAmount is empty

DO

SET Warning popup " Dear User, We kindly request that you fill out the form that has been given to you."

ENDDO

ELSEIF

DO

TRY

DO

SET text int debitCard_balanceAmount as Integer parse Int of

debit_balanceAmount

SET text int debitCard_cardID as Integer parse Int of debit_cardID

SET text int debitCard_PIN_Number as Integer parse Int of debit_PIN_Number

SET text String debitCard_clientName as debit_clientName

SET text String debitCard_bankAccount as debit_bankAccount

SET text String debitCard_issuerBank as debit_issuerBank

IF the list is empty

DO

SET DebitCard Debit_Parameters as new

DebitCard(debitCard_balanceAmount,debitCard_cardID,debitCard_bankAccount,debitCard_issu

erBank,debitCard_clientName,debitCard_PIN_Number)

ADD GUI Debit_Parameters

ENDIF

ENDDO

ELSEIF

DO

SET BankCard card in GUI

DO

IF card instance of DebitCard

DO

SET DebitCard debitCard as (DebitCard) card

IF debitCard.getcardId() == Debit_cardID

DO

SET popup “The addition of a Debit Card to the account has been confirmed "

ENDIF

ENDDO

ELSEIF

DO

SET DebitCard Debit_Parameters as new DebitCard(

debitCard_balanceAmount,debitCard_cardID,debitCard_bankAccount,debitCard_issuerBank,de

bitCard_clientName,debitCard_PIN_Number)

ADD GUI Debit_Parameters

SET popup " The addition of the Debit Card was successful. "

ENDDO

ENDIF

ENDDO

ENDDO

ENDDO

ENDDO

CATCH Exception

DO

SET warning popup, " Kindly review the provided form and ensure that all information has been accurately provided. "

ENDDO

ENDDO

ENDIF

ENDDO

IF (e.getSource() == Display button for Debit Card)

DO

IF (Debit_clientName_T is empty, Debit_issuerBank_T is empty, Debit_bankAccount_T is empty, Debit_cardID_T.getText().is empty, Debit_balanceAmount_T is empty, Debit_pinNumber_T is empty)

DO

SET warning popup "Please fill out the form with the necessary data"

ENDIF

ENDDO

ELSEIF

DO

TRY

DO

ENDDO

CATCH Exception

DO

SET warning popup "Kindly review the provided form and ensure that all information has been accurately provided.

ENDDO

ENDDO

SET BankCard Debit in list

DO

IF a Debit instance of DebitCard

DO

DISPLAY DebitCard Debit

ENDIF

ENDDO

ENDIF

ENDDO

DO

IF (e.getSource() == button for Clear Debit Card)

DO

SET TEXT debit_clientName, debit_issuerBank , debit_bankAccount debit_cardID , debit_PIN_Number and debit_balanceAmount to empty

ENDDO

ENDIF

ENDDO

DO

IF (e.getSource() == button for Clear Withdrawal)

DO

SET TEXT withdraw_amount_T , withdraw_cardID_T , withdraw_PIN_Number_T to empty

ENDDO

ENDIF

ENDDO

DO

IF (e.getSource() == button for Clear Credit)

DO

SET TEXT creditCard_clientName_T , creditCard_cardID_T creditCard_issuerBank_T , creditCard_interestRate_T , creditCard_bankAccount_T creditCard_balanceAmount_T, creditCard_CVCNumber_T to empty

ENDDO

ENDIF

ENDDO

IF (e.getSource() == button for Clear Credit limit)

DO

SET TEXT creditCard_cardID_t, creditCard_gracePeriod_T, creditCard_creditLimit_T, to empty

ENDDO

ENDIF

ENDDO

IF (e.getSource() == button for clear Credit card)

DO

SET TEXT cancel_cardID_T, to empty

ENDDO

ENDIF

ENDDO

5. Method Description

A method is a sequence of instructions or statements that are used in Java programming to carry out a particular activity. A method can be specified with access modifiers like public, private, or protected and have a name, input arguments, and a return value. The method's name ought to be evocative and reflect the function it fulfills. A method's name and the necessary input arguments can be passed when calling it from another area of the program. To ensure that the code is efficiently maintained and that other developers can understand how to use the methods appropriately, it is crucial to write clear and concise method descriptions.

5.1 Bank GUI Methods

The Bank GUI consists of two methods. They are

5.1.1 Main Method (String [] args)

The main method is a program's entry point in Java programming. It is a unique method that is invoked as soon as the application runs. A special signature for the main method includes the keywords "public," "static," and "main." Additionally, it contains a String array parameter that is used to pass program arguments on the command line. The main method is in charge of initializing the program's state, generating objects, and calling additional methods as required. It is frequently used to read user input, initialize variables, and carry out other operations necessary to launch the program's execution. The main method is a crucial part of every Java application and needs to be present in every class that is designed to be executable.

5.1.2 actionPerformed(ActionEvent e)

In Java, the actionPerformed(ActionEvent e) method handles events that occur when a user interacts with a graphical user interface (GUI) component, such as a button or a menu item. This function belongs to the ActionListener interface and is usually implemented by a class that listens for events created by a certain component. When an event occurs, such as a button click, the actionPerformed() method is invoked and an ActionEvent object is passed as an argument. This object holds event information such as the event's sender and any user input related with the event. The actionPerformed() method is in charge of responding to the event, which can include updating the GUI's state, conducting calculations, or launching a procedure. This method is a key aspect of Java GUI programming, and knowing how to utilize it properly is critical for developing responsive and user-friendly programs.

6 Testing

We run the test to determine if the code is functioning properly.

6.1 Test Case 1

Objective	Using the command prompt, see if the program can be compiled and run.
Action	Start the command prompt, navigate to the file location, and run the application.
Expected Result	The application will execute in the command prompt and display the graphical user interface.
Actual Result	The application was executed in the command prompt and the graphical user interface was displayed.
Conclusion	The program run was completed successfully.

Table 1 Test table 1

6.1.1 Table Test 1 (Screenshots)

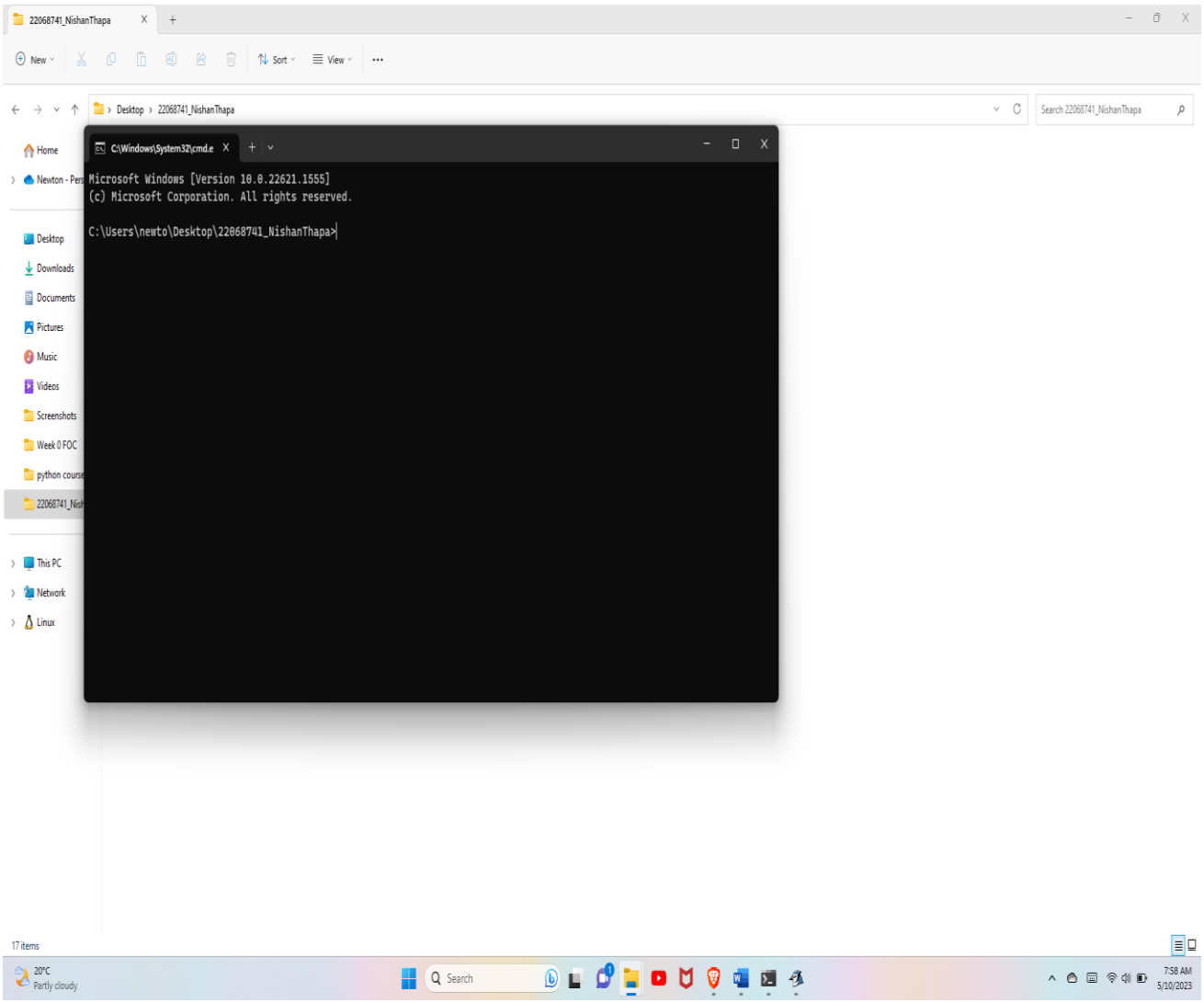

Figure 10 screenshot of test1

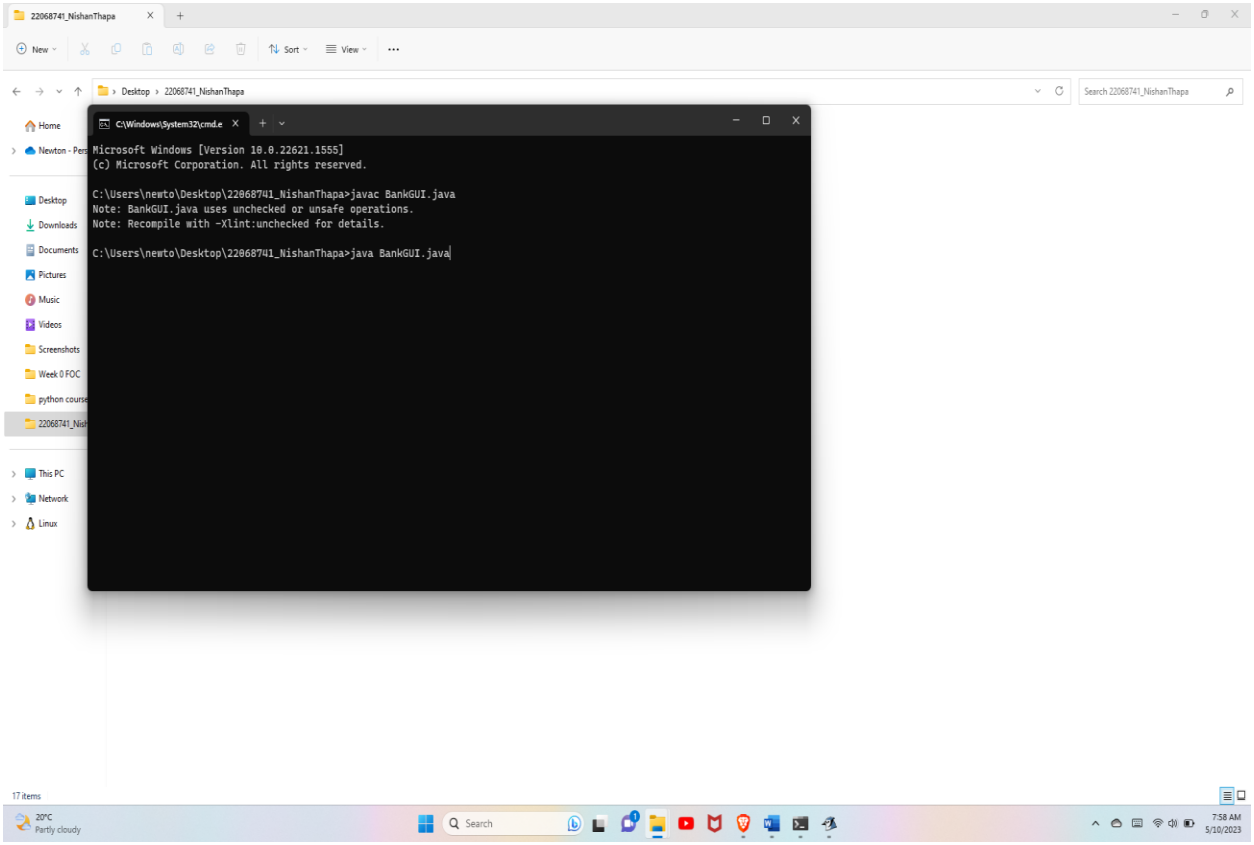

Figure 11 screenshot of test 1

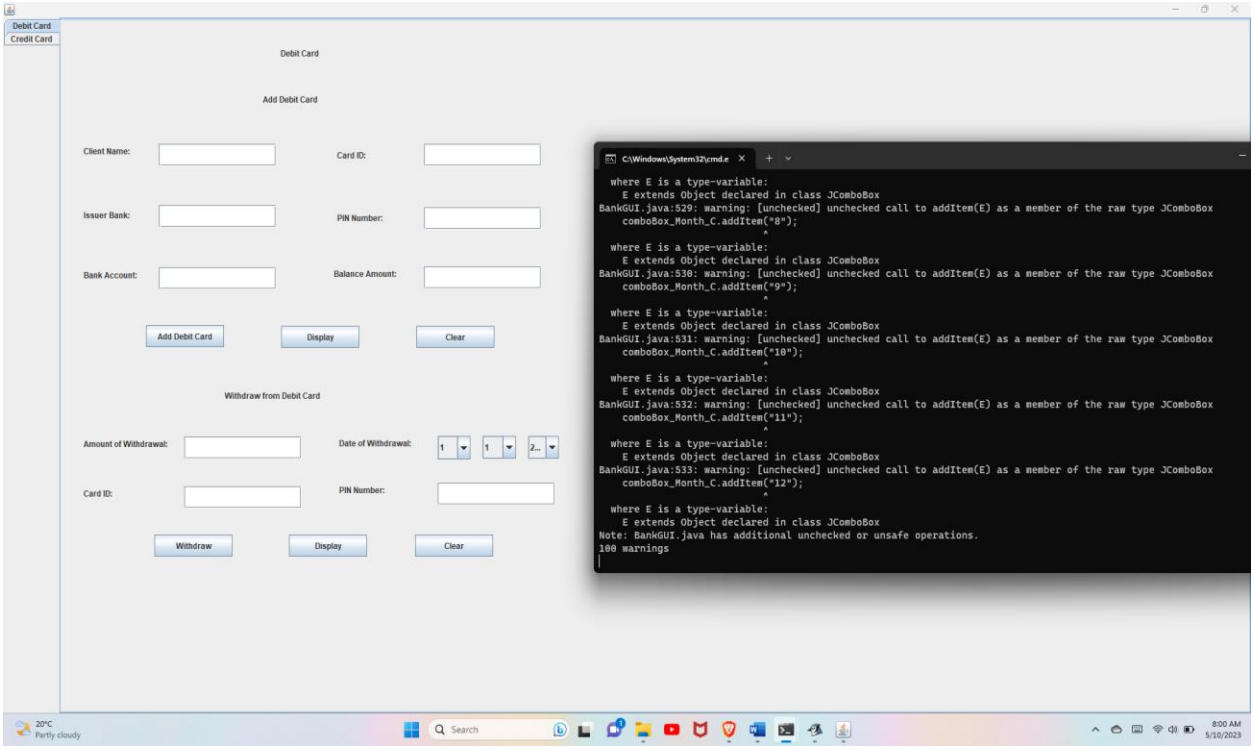

Figure 12 screenshot of test 1

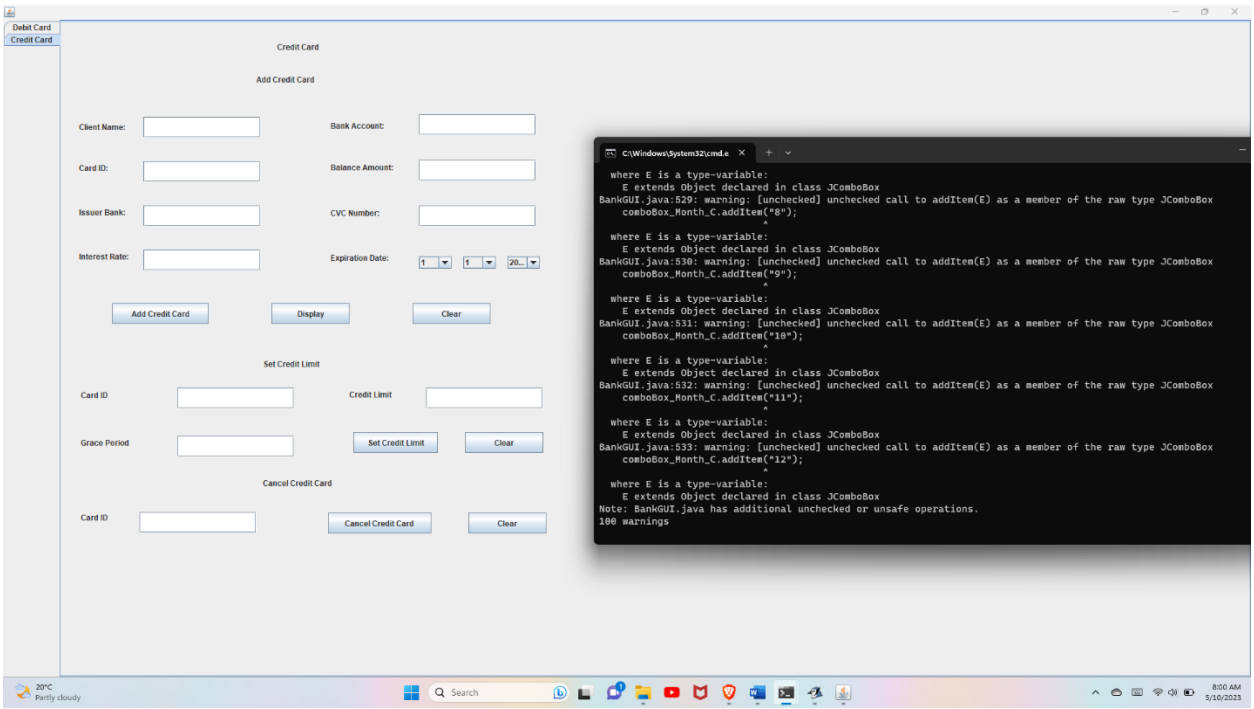

Figure 13 screenshot of test 1

6.2 Test Case 2 (a) (Add Debit Card Table)

Objective	To add the objects of Debit Card
Action	The following parameters were filled in the debit card: Client Name – Nishan Thapa Issuer Bank – Nabil Bank Account – 0J600 Card ID – 07 PIN Number – 5756 Balance Amount - 1000
Expected Result	A pop-up notice should display on the screen after the debit card is added.
Actual Result	A pop-up notice was displayed on the screen after the debit card was added.
Conclusion	The test was successful

Table 2 Test Table 2

6.2.1 Table Case 2 (a) Screenshots

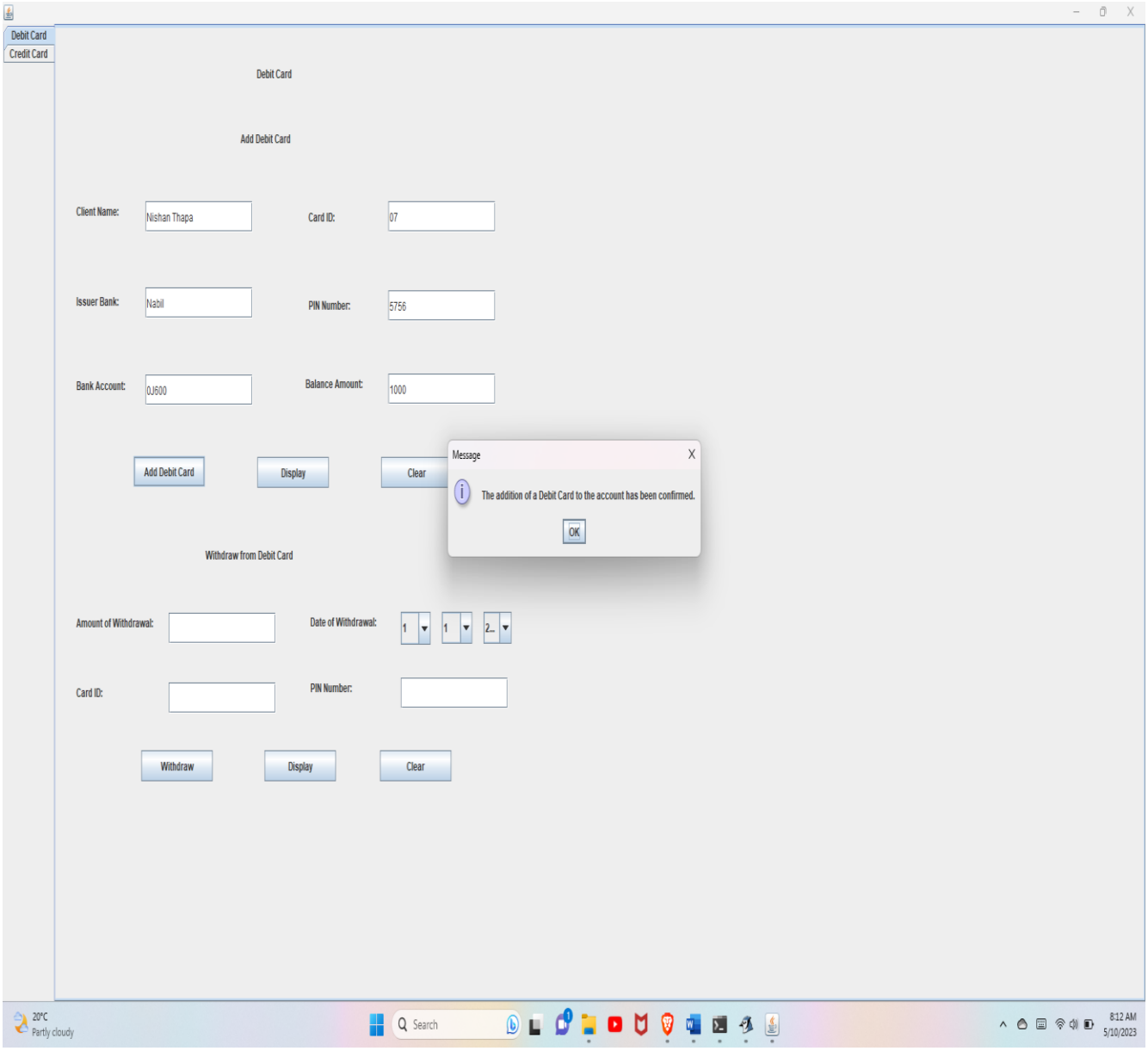

Figure 14screenshot of test 2

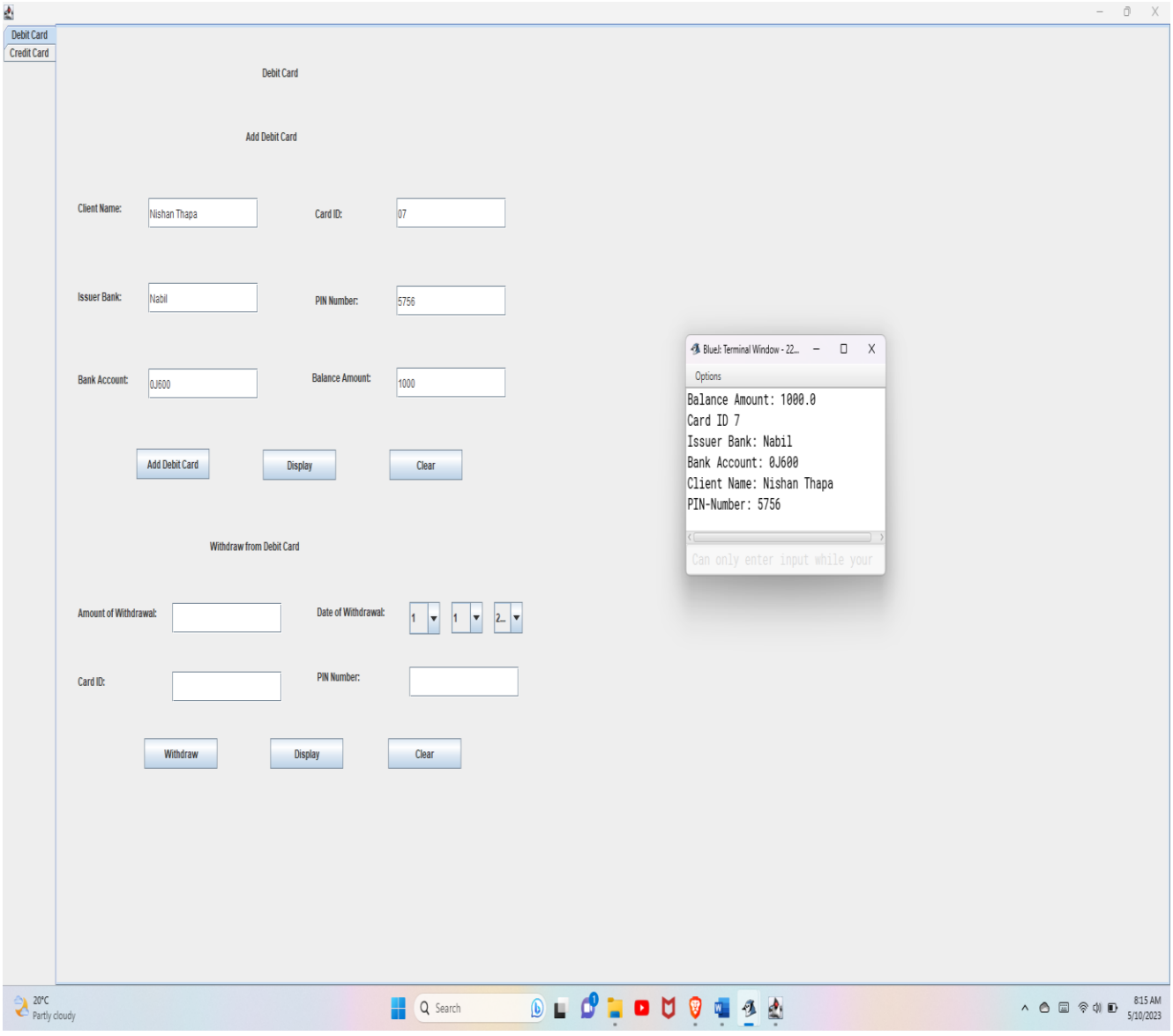

Figure 15 screenshot of test 2

6.3 Test Case 2 (b) (Add of credit card)

Objective	To add the objects of Credit Card
Action	The following parameters were filled in the Credit card: Client Name – Nishan Thapa Issuer Bank – Nabil Bank Account – 0J600 Card ID – 07 Interest Rate – 13 Balance Amount – 1000 CVC Number – 576 Expiration Date – 10/5/2023
Expected Result	A pop-up notice should display on the screen after the credit card is added.
Actual Result	A pop-up notice was displayed on the screen after the credit card was added.
Conclusion	The test was successful

Table 3 Test 3

6.3.1 Table Case 2 (b) (Screenshots)

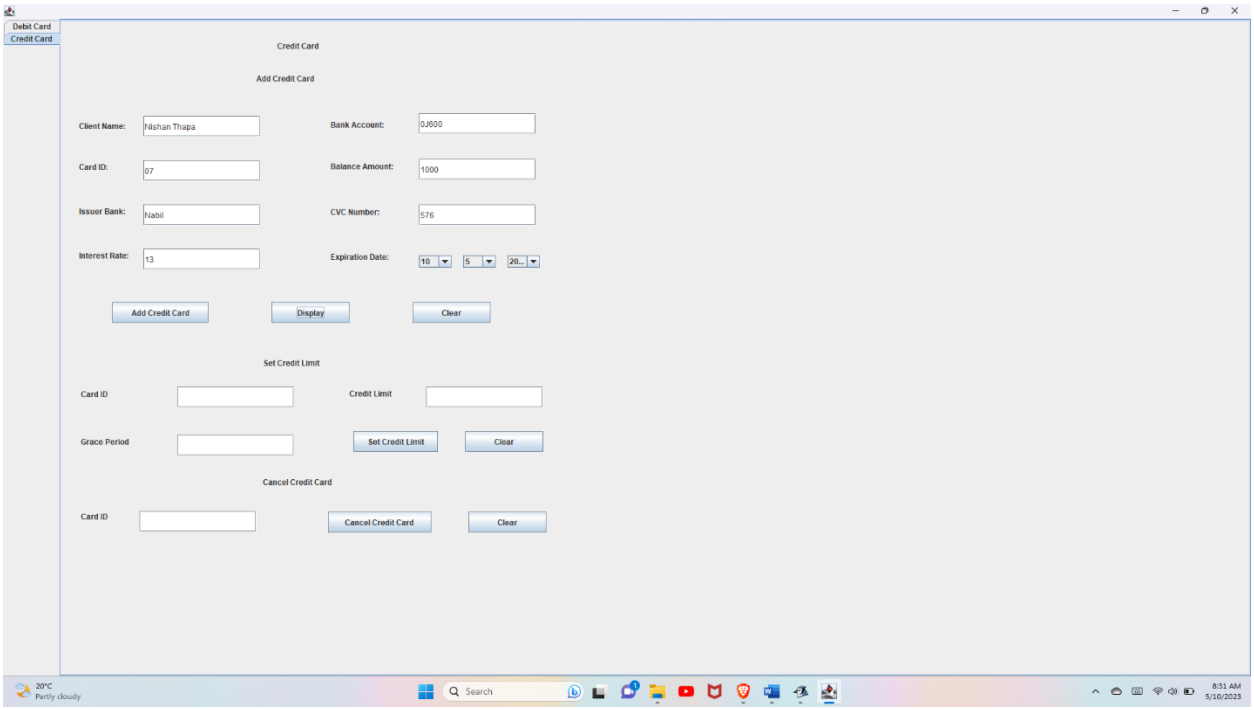

Figure 16 screenshot of test 3

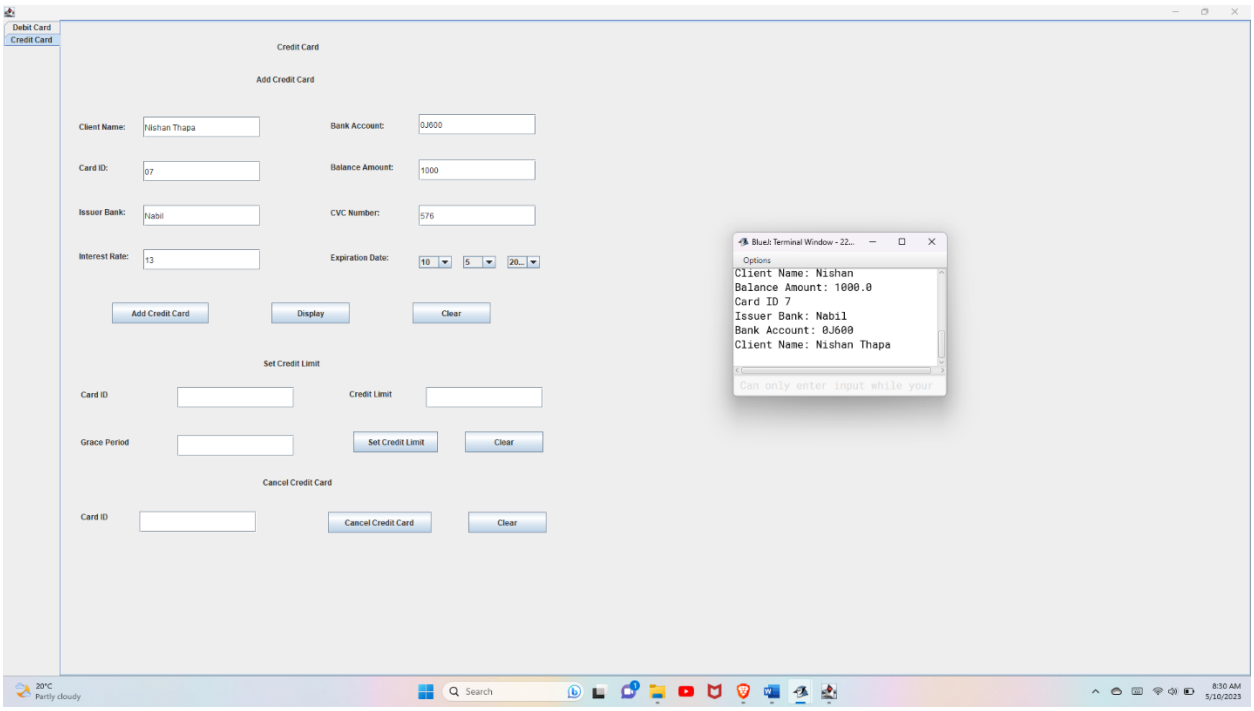

Figure 17 screenshot of test 3

6.4 Test case 2 (c) (Withdrawal Table)

Objective	To withdraw amount from Debit Card
Action	The following parameters were filled in the withdrawal of debit card: Withdrawal Amount – 500 Date of Withdrawal – 10/05/2023 Card ID – 07 PIN Number – 5756
Expected Result	A pop-up notice should display on the screen after the withdrawal.
Actual Result	A pop-up notice was displayed on the screen after the withdrawal was successful.
Conclusion	The test was successful

Table 4 Test 4

6.4.1 Table Case 2 (c) Screenshots

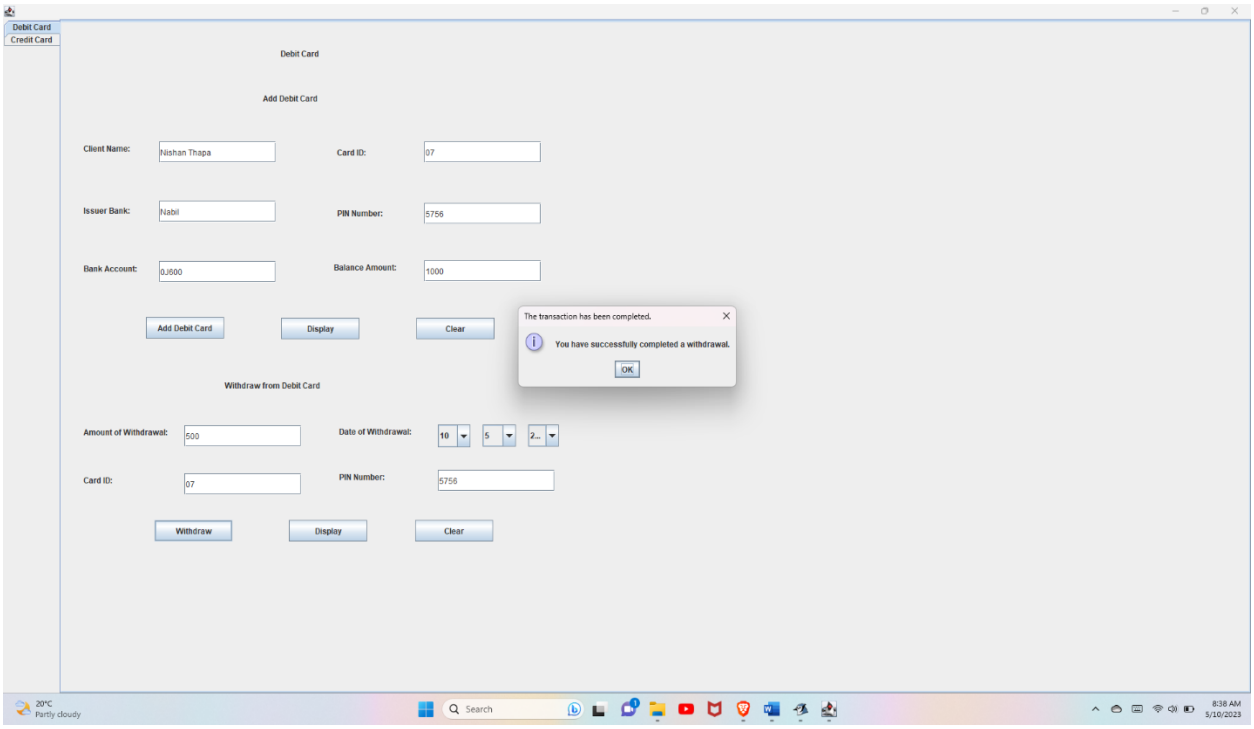

Figure 18 screenshot of test 4

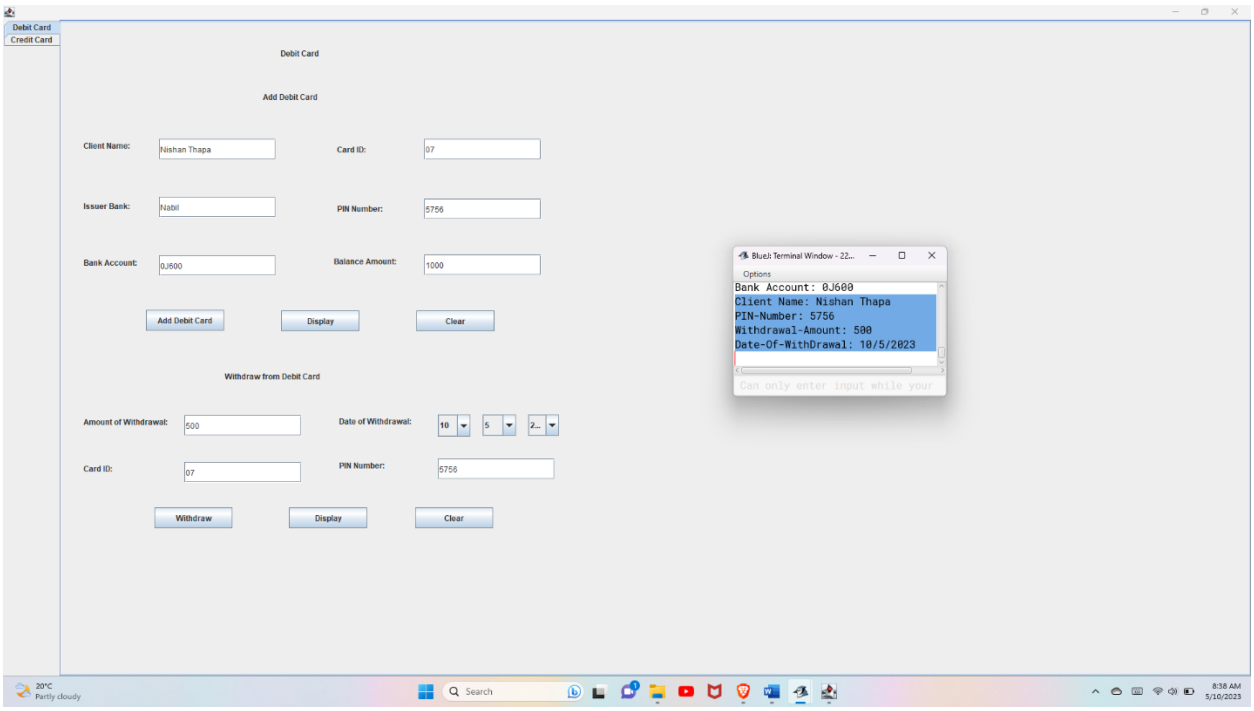

Figure 19 screenshot of test 4

6.5 Test case 2 (d) Credit Limit

Objective	To set the credit limit of the credit card
Action	The following parameters were filled in the credit card: Credit Limit – 20 Grace Period – 10
Expected Result	A pop-up notice should display on the screen after the credit limit is set.
Actual Result	A pop-up notice was displayed on the screen after the credit limit was set.
Conclusion	The test was successful

Table 5 Test 5

6.5.1 Test Case 2 (d) Screenshots

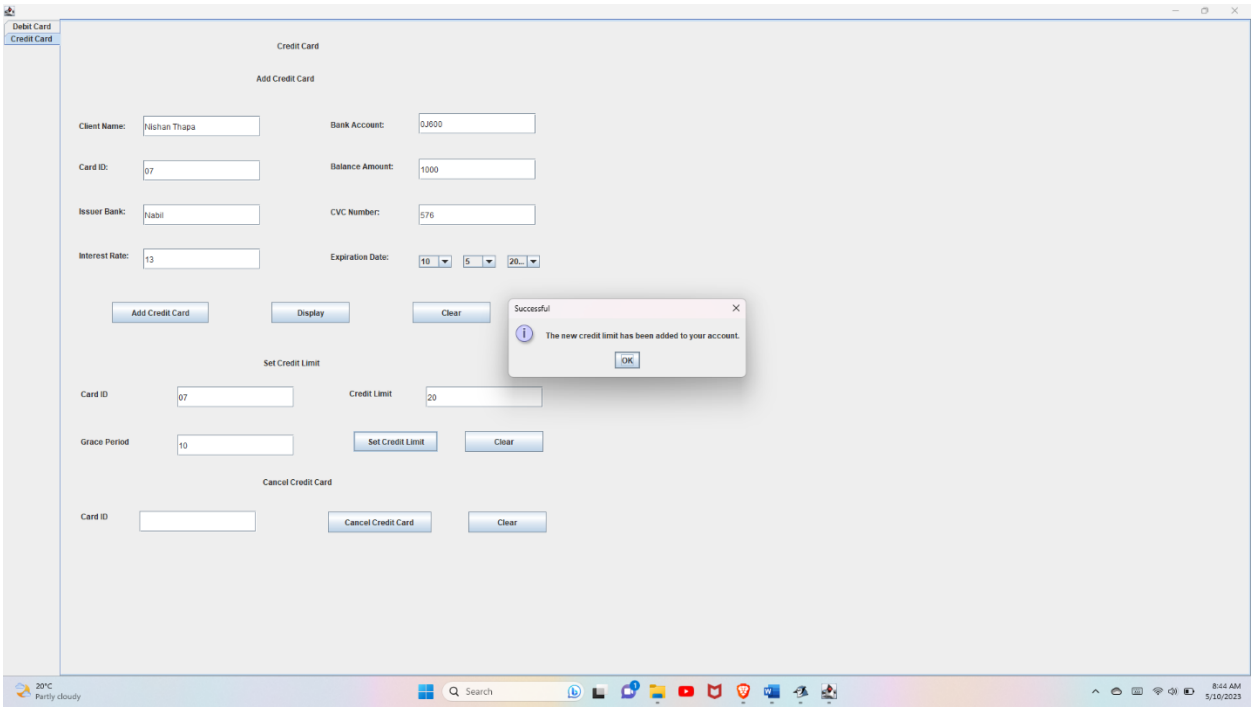

Figure 20 screenshot of test 5

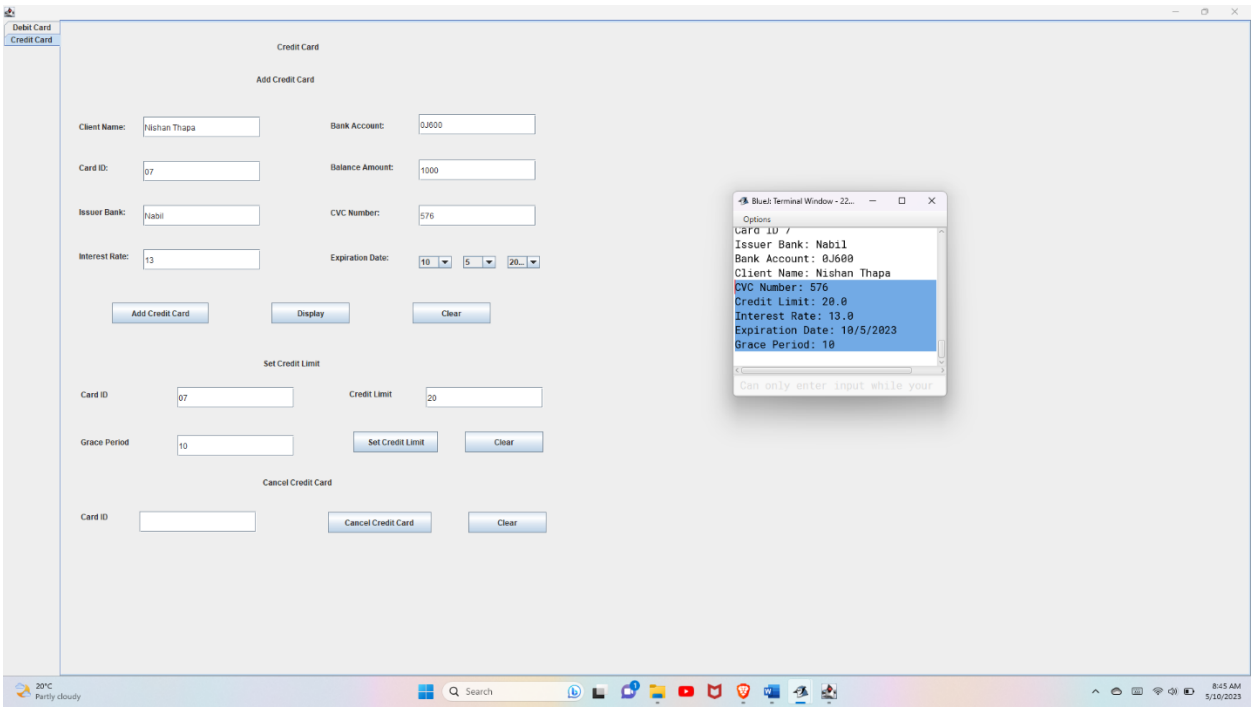

Figure 21 screenshot of test 5

6.6 Test Case 2 (e) Cancel Credit Card

Objective	To cancel the credit card
Action	The following parameters were filled in the cancel credit card: Card ID – 07
Expected Result	A pop-up notice should display on the screen after the credit card is cancelled.
Actual Result	A pop-up notice was displayed on the screen after the credit card was cancelled.
Conclusion	The test was successful

Table 6 Test 6

6.6.1 Test Case 2 (e) Cancel Credit Card Screenshots

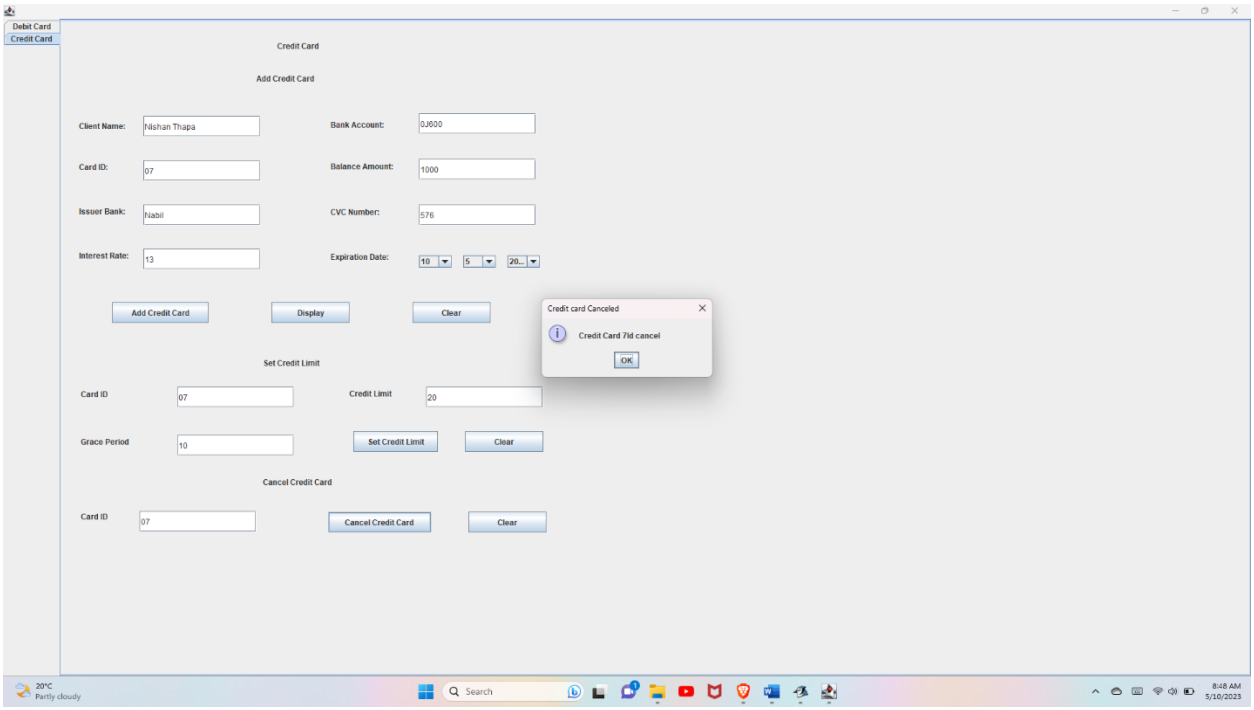

Figure 22 screenshot of test 6

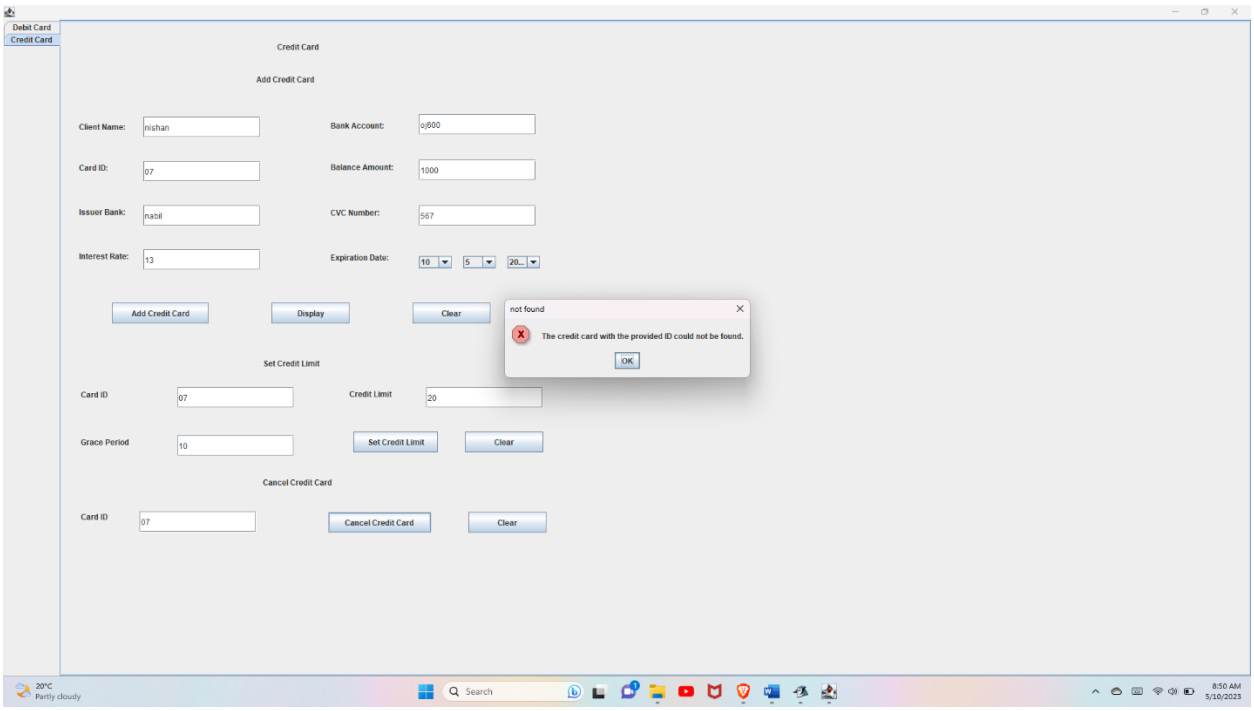

Figure 23 screenshot of test 6

6.7 Test Case 3 (a) Testing for unmatched Card ID's

Objective	To check whether the program runs when unmatched Card ID input is provided by the user.
Action	The following parameters were filled in the Credit card section: Client Name – Nishan Thapa Issuer Bank – Nabil Bank Account – 0J600 Card ID – 07 Interest Rate – 13 Balance Amount – 1000 CVC Number – 576 Expiration Date – 10/5/2023 Set Credit Limit Credit Limit – 20 Grace Period – 10 Card ID – 10 (unmatched)

Expected Result	A pop-up notice should display on the screen saying card ID does not match.
Actual Result	A pop-up notice was displayed on the screen.
Conclusion	The test was successful

Table 7 Test 7

6.7.1 Test Case 3 (a) Testing for unmatched Card ID's Screenshots

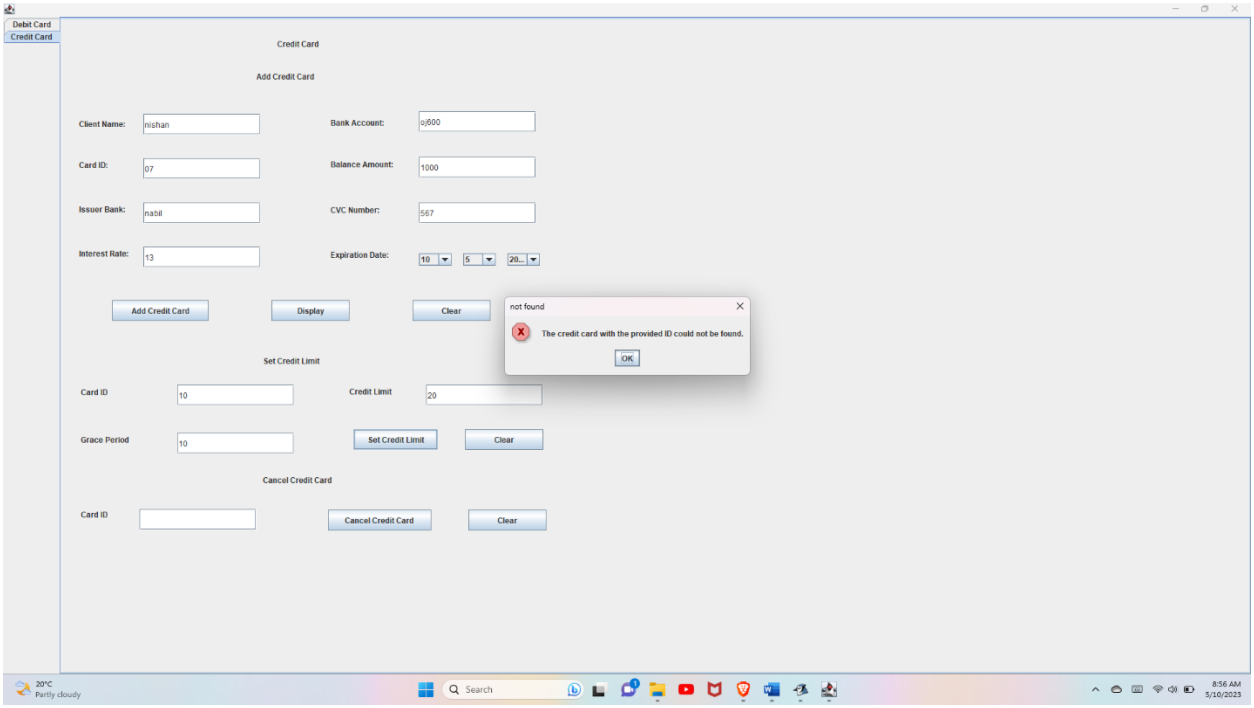

Figure 24 screenshot of test 7

6.8 Test Case 3 (b) Testing for unmatched PIN Numbers

Objective	To check whether the program runs when unmatched PIN Number input is provided by the user.
Action	The following parameters were filled in the debit card: Client Name – Nishan Thapa Issuer Bank – Nabil Bank Account – 0J600 Card ID – 8 PIN Number – 5756 Balance Amount – 1000 Withdraw Debit Card Withdrawal Amount – 500 Card ID – 8 Date of Withdrawal – 1/1/2013 PIN Number - 5656
Expected Result	A pop-up notice should display on the screen saying the PIN Number does not match.

Actual Result	A pop-up notice was displayed on the screen.
Conclusion	The test was successful

Table 8 Test 8

6.8 Test Case 3 (b) Testing for unmatched PIN Numbers Screenshots

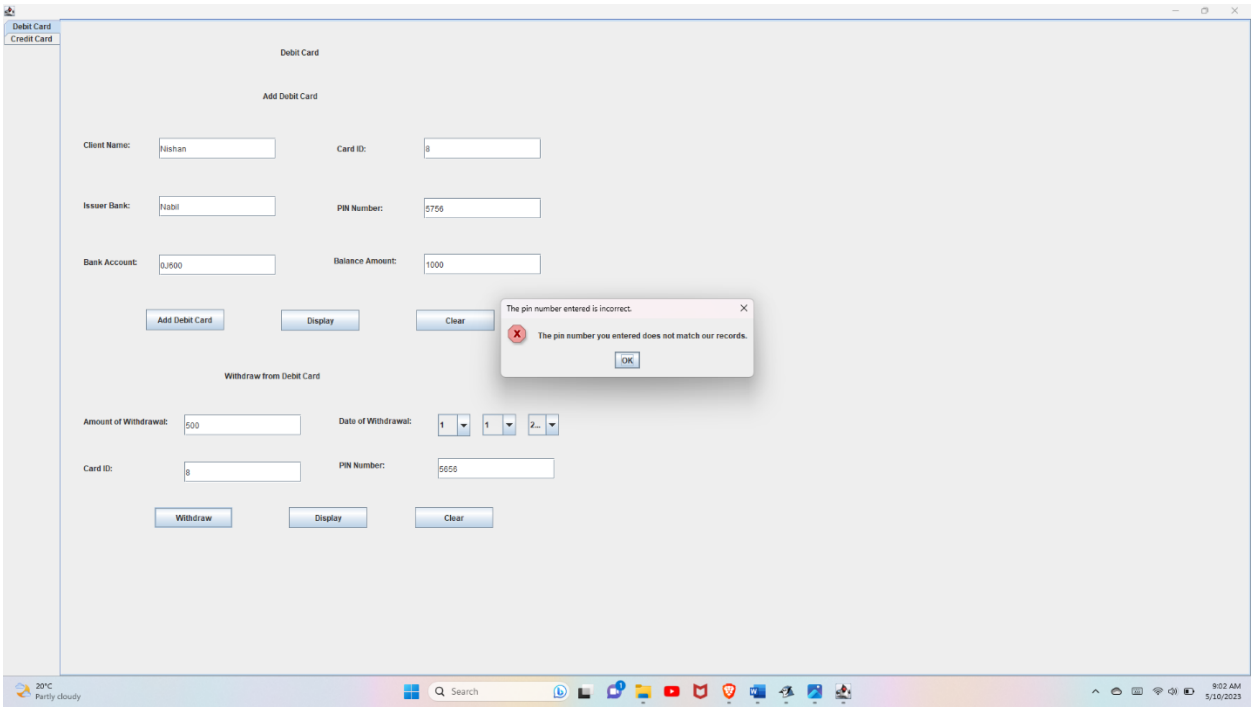

Figure 25 screenshot of test 8

6.8.1 Test Case 3 (c) Testing for PIN Numbers more than four

Objective	To check if the GUI excepts more than four digit numbers as PIN Number.
Action	The following parameters were added to debit card Client Name – Nishan Thapa Issuer Bank – Nabil Card ID – 8 Bank Account – 0J600 Balance Amount – 1000 PIN Number – 57566 (5 digits)
Expected Result	A pop-up message saying PIN number should be less than 4.
Actual Result	A pop-up message was displayed on the screen.
Conclusion	The Test was successful.

Table 9 Test 9

6.8.1 Test Case 3 (c) Testing for PIN Numbers more than four Screenshots

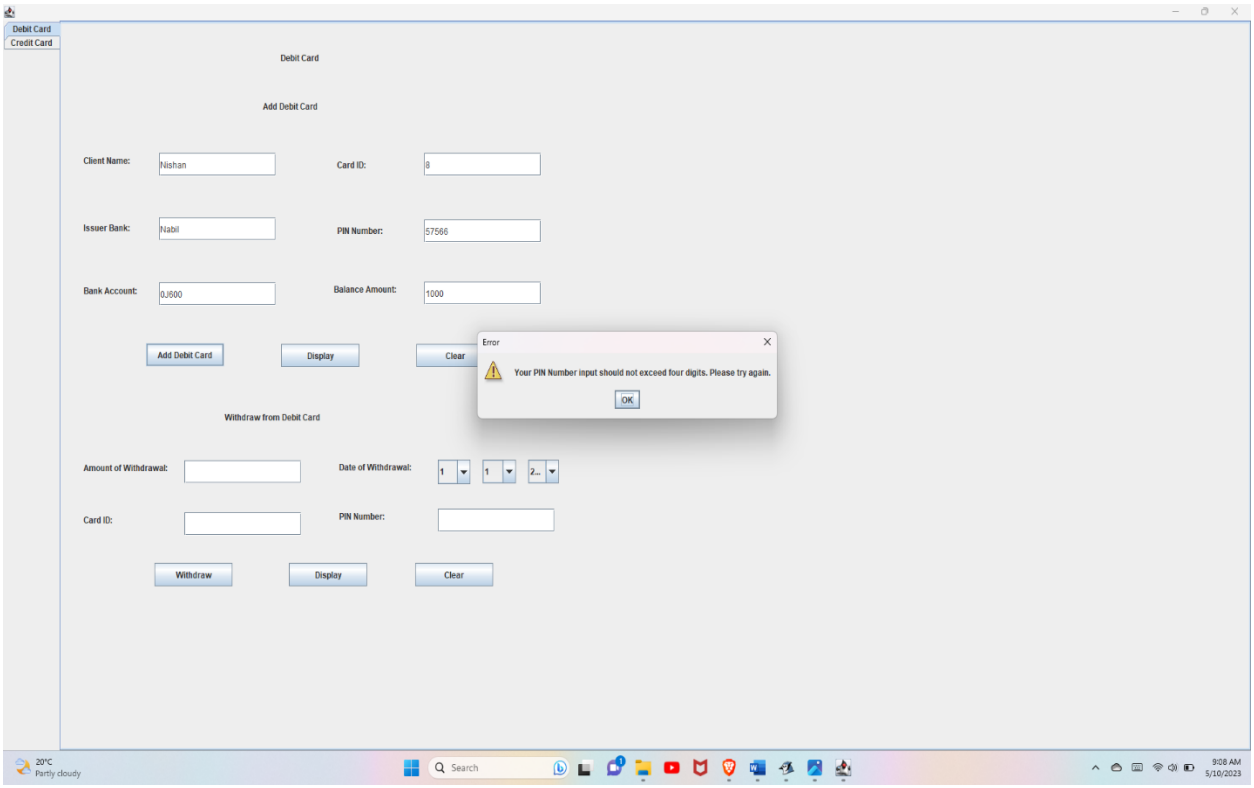

Figure 26 screenshot of test 9

7 Test Case 3 (d) Testing for when add button is pressed without filling the form.

Objective	To check if the form is added with no parameters provided by the user.
Action	The following parameters were filled in the Debit card: Client Name – Issuer Bank – Bank Account – Card ID – PIN Number – Balance Amount - The following parameters were filled in the Credit card: Client Name – Issuer Bank – Bank Account – Card ID – Interest Rate –

	Balance Amount – CVC Number – Expiration Date –
Expected Result	A pop-up notice should display on the screen saying fill the form with necessary information.
Actual Result	A pop-up notice was displayed on the screen.
Conclusion	The test was successful

Table 10 Test 10

7.1 Test Case 3 (d) Testing for when add button is pressed without filling the form Screenshots.

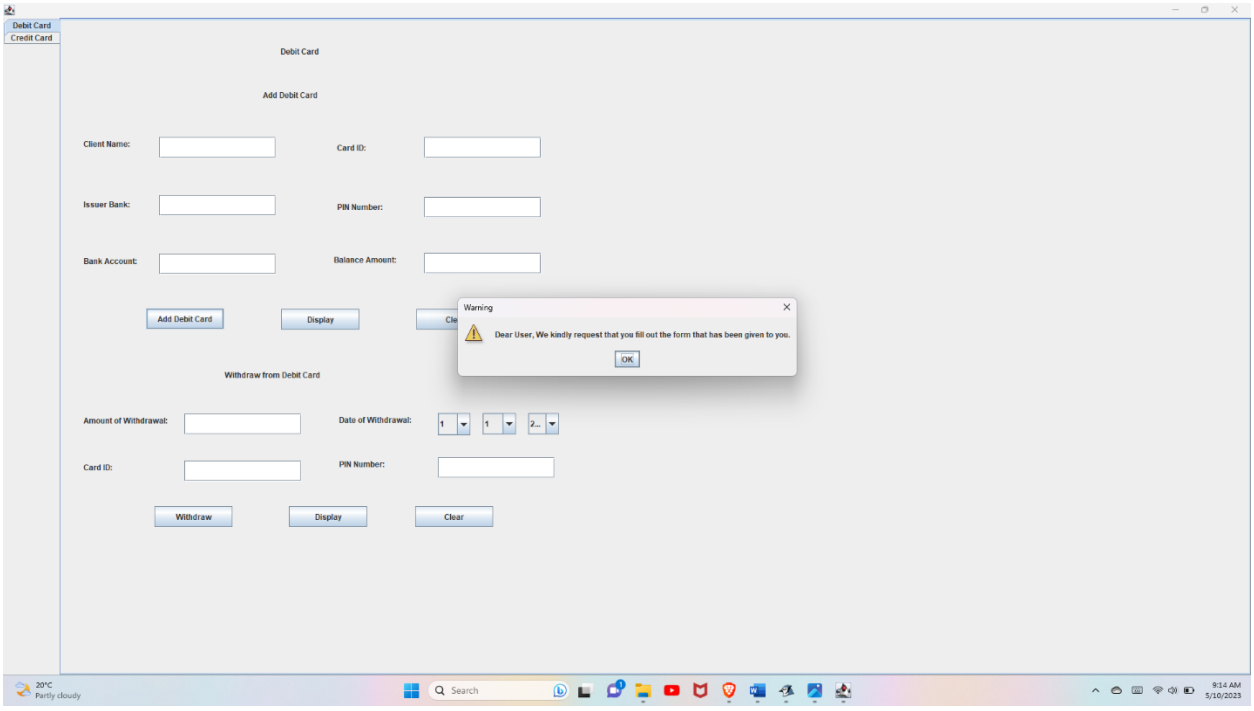

Figure 27 screenshot of test 10

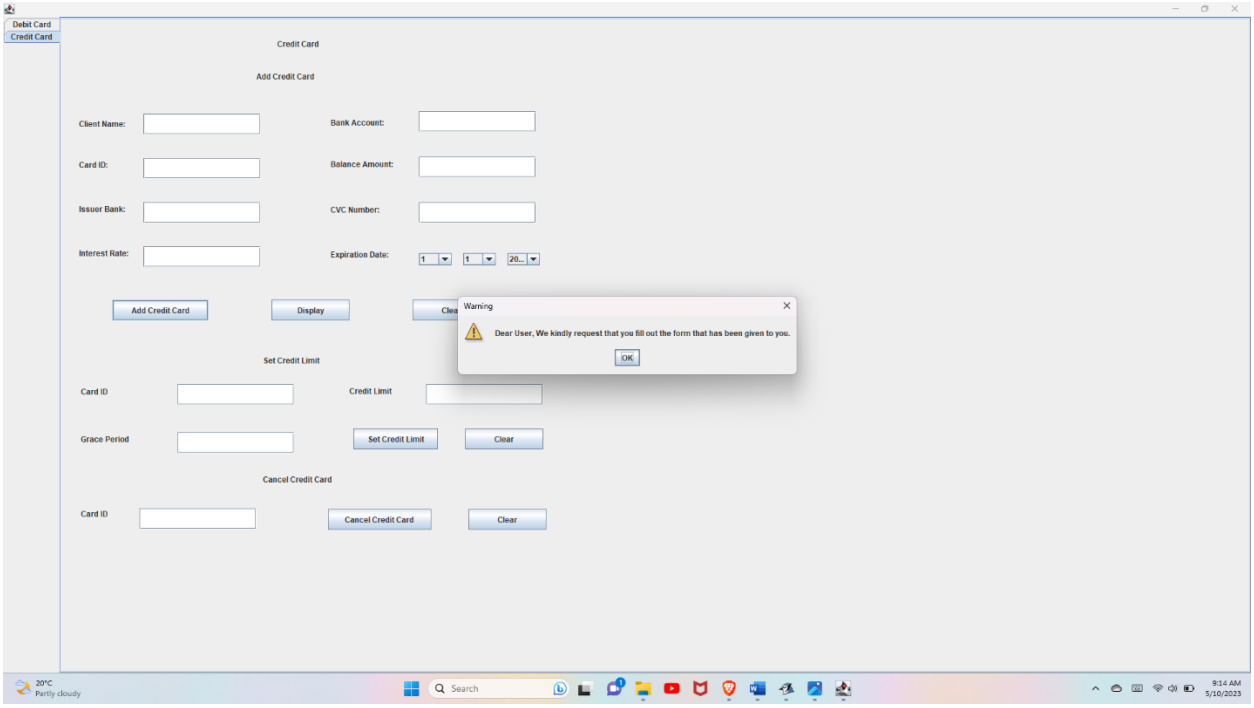

Figure 28 screenshot of test 10

7.2 Test Case 3 (e) Testing for when add button is pressed when few section of form is left empty.

Objective	To check if the form is added with few of the parameters provided by the user is left empty.
Action	The following parameters were filled in the Debit card: Client Name – Nishan Thapa Issuer Bank – Nabil Bank Account – 0J600 Card ID – PIN Number – 5756 Balance Amount - 1000 The following parameters were filled in the Credit card: Client Name – Nishan Thapa Issuer Bank – Nabil Bank Account – 0J600 Card ID –

	Interest Rate – 13 Balance Amount – 1000 CVC Number – 576 Expiration Date – 1/1/2013
Expected Result	A pop-up notice should display on the screen saying fill the form completely.
Actual Result	A pop-up notice was displayed on the screen.
Conclusion	The test was successful

Table 11 Test 11

7.2.1 Test Case 3 (e) Testing for when add button is pressed when few section of debit form is left empty. Screenshots

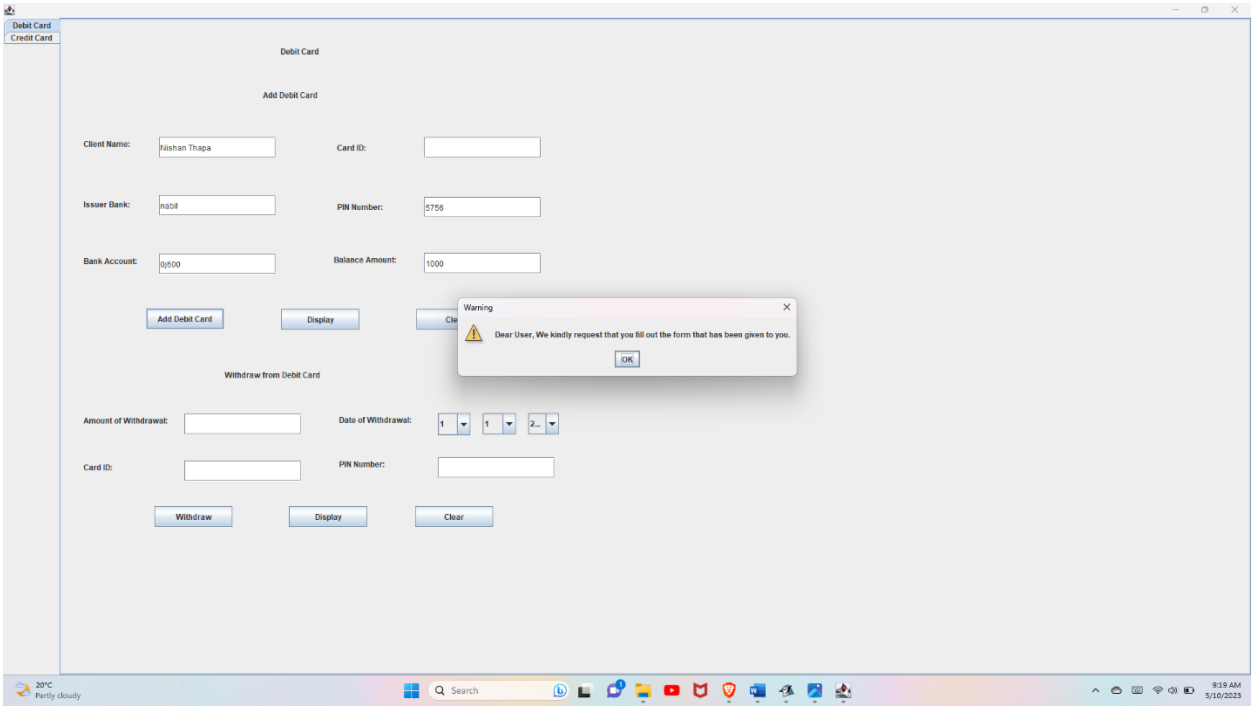

Figure 29 screenshot of test 11

7.2.2 Test Case 3 (e) Testing for when add button is pressed when few section of credit form is left empty. Screenshots

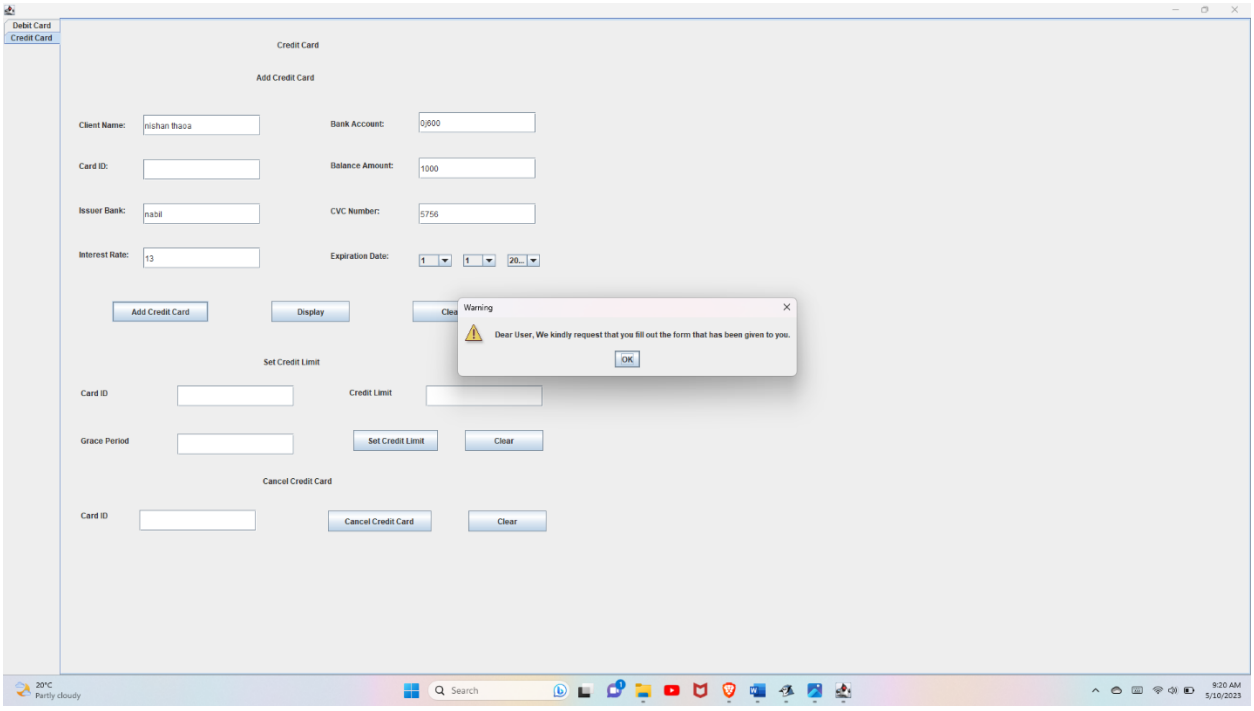

Figure 30 screenshot of test 11

8 Error detection and corrections

Java programming relies heavily on error detection and testing. Java has try-catch blocks, exceptions, and debugging tools such as the Java Virtual Machine Debugger (JVM Debugger) and the Eclipse Integrated Development Environment (IDE) for identifying and handling failures. The try-catch block handles exceptions that may arise during program execution, whereas the exception handling mechanism allows for the generation and management of unique exceptions.

8.1 Syntax Error

A syntax error in Java occurs when the code violates the rules of the Java programming language syntax. It is a type of error that is caught by the compiler during the compilation process. A syntax error is usually caused by incorrect usage of Java keywords, missing or extra brackets or semicolons, misspelled method or variable names, or an incorrect order of arguments in a method call.

8.1.1 Error Table

Error Number	1
Error Type	Syntax
Error	Variable Names were similar
Correction	Variable Name was changed
Conclusion	Should use different variables for Labels, Text Fields and Buttons.

Table 12 Error Table 1

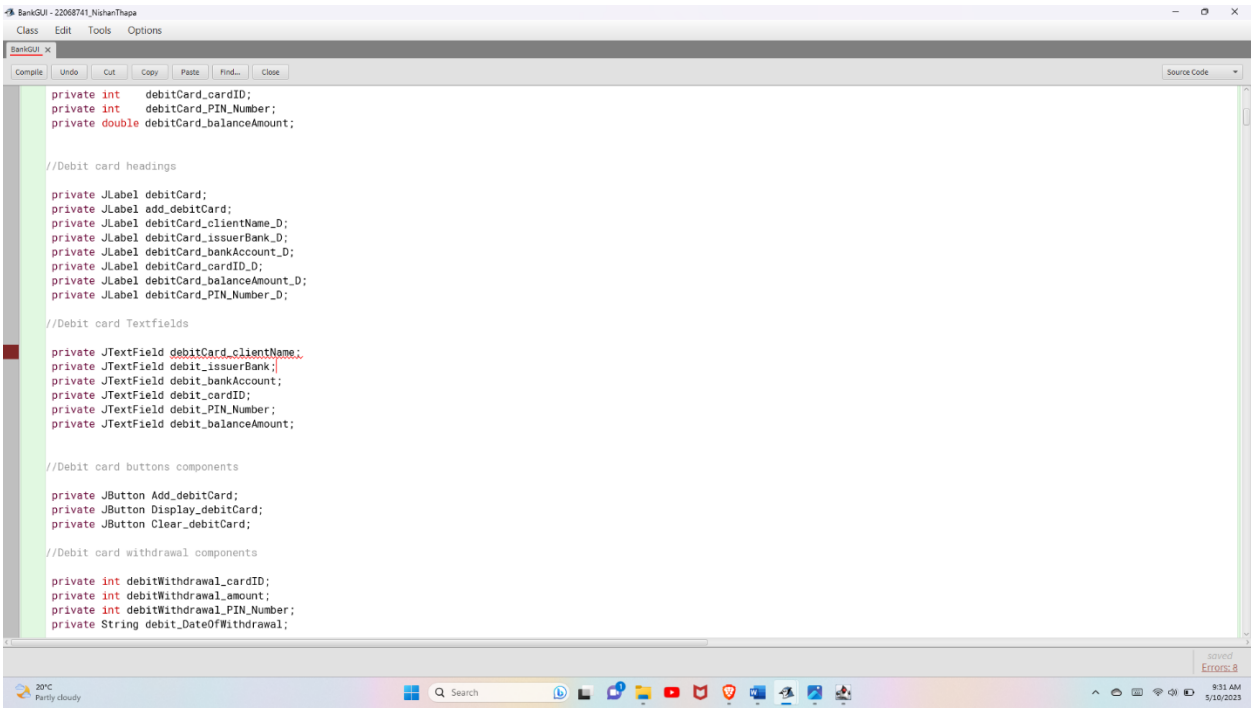

Figure 31 screenshot of error 1

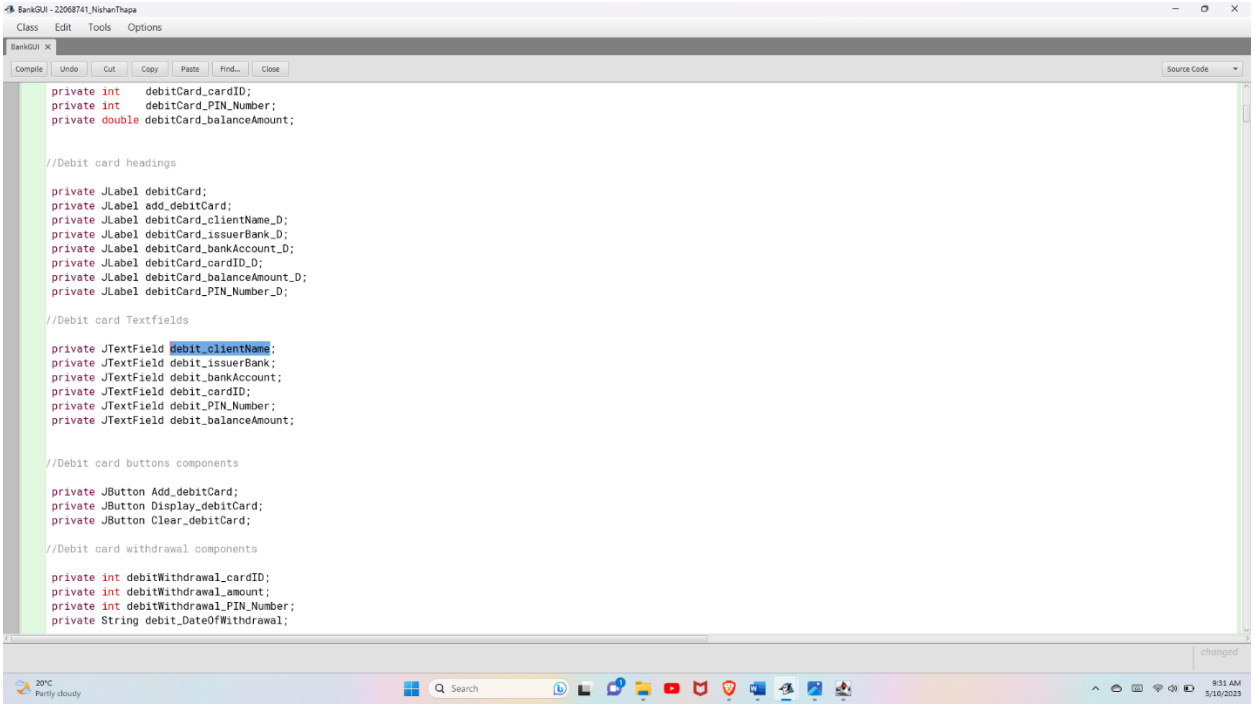

Figure 32 screenshot of error 1

8.2 Semantic errors

A semantic error in Java occurs when the code runs without any compilation errors, but it produces incorrect or unexpected results at runtime. Semantic errors are also known as logical errors because they involve errors in the logical flow of the program. These types of errors are typically more difficult to detect than syntax errors because the code can be executed without any visible issues, but it produces incorrect results.

8.2.2 Semantic errors

Error Number	2
Error Type	Semantic
Error	Variables were declared public
Correction	Variable were changed to private
Conclusion	Should use private variables for Labels, Text Fields and Buttons etc. for safety.

Table 13 error table 2

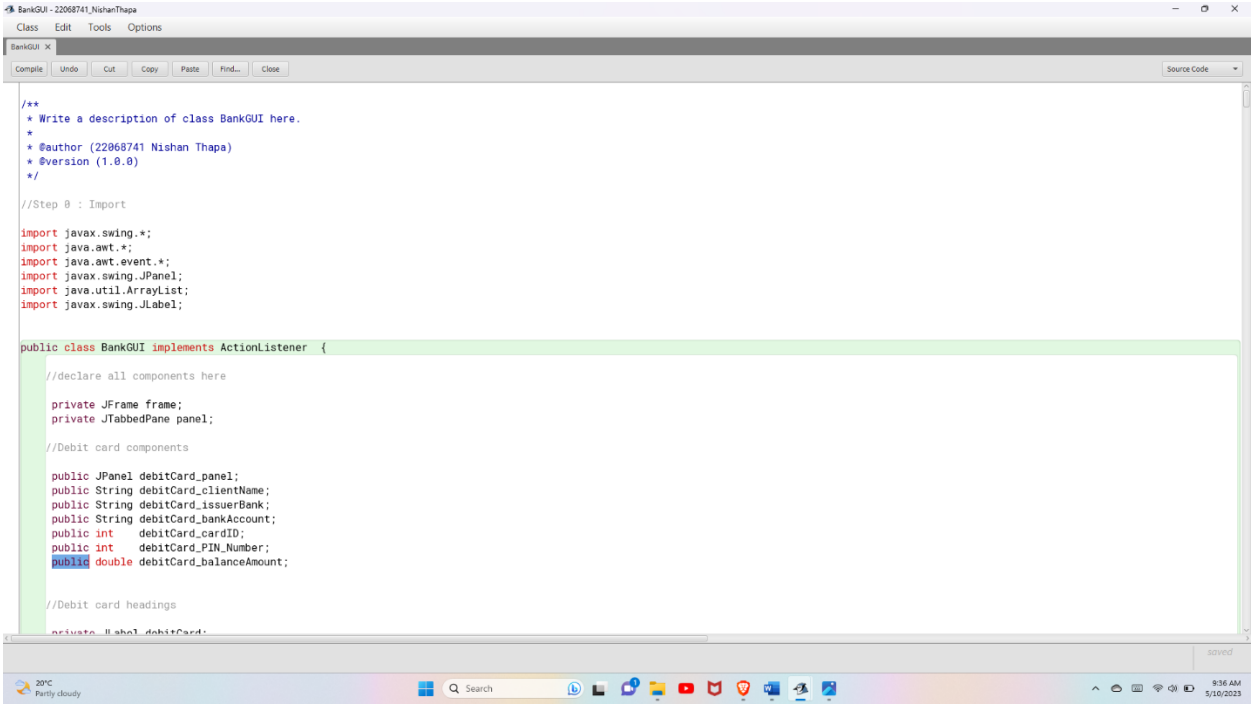

Figure 33 screenshot of error 2

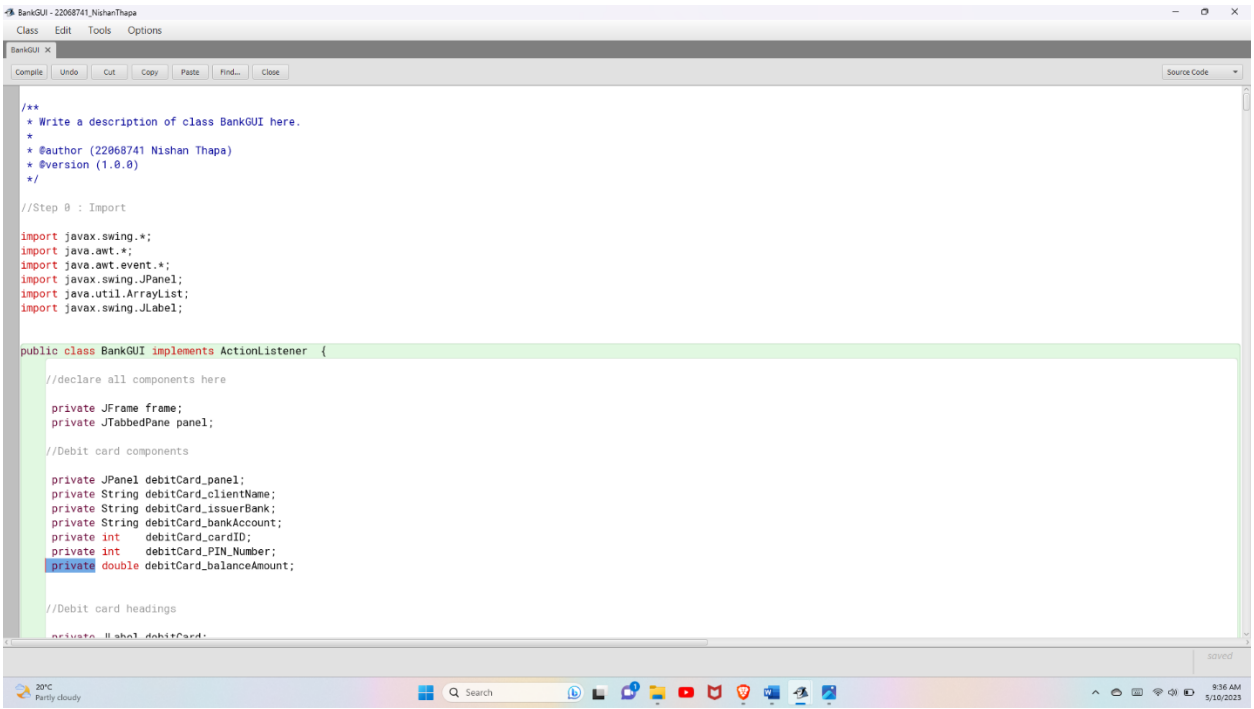

Figure 34 screenshot of error 2

8.3 Logical Errors

Logical mistakes, arise in software when code is syntactically correct but delivers unexpected effects owing to a flaw in its logic. Logical mistakes are frequently caused by faulty assumptions or misconceptions of the program's requirements or objectives.

8.3.1 Logical Error Table

Error Number	3
Error Type	Logical
Error	When card ID matches for add and withdraw debit card the Boolean value for has Withdrawn was set false.
Correction	When card ID matches the Boolean value for has Withdrawn was set true.
Conclusion	Assumptions should be clear.

Table 14 Error Table 3

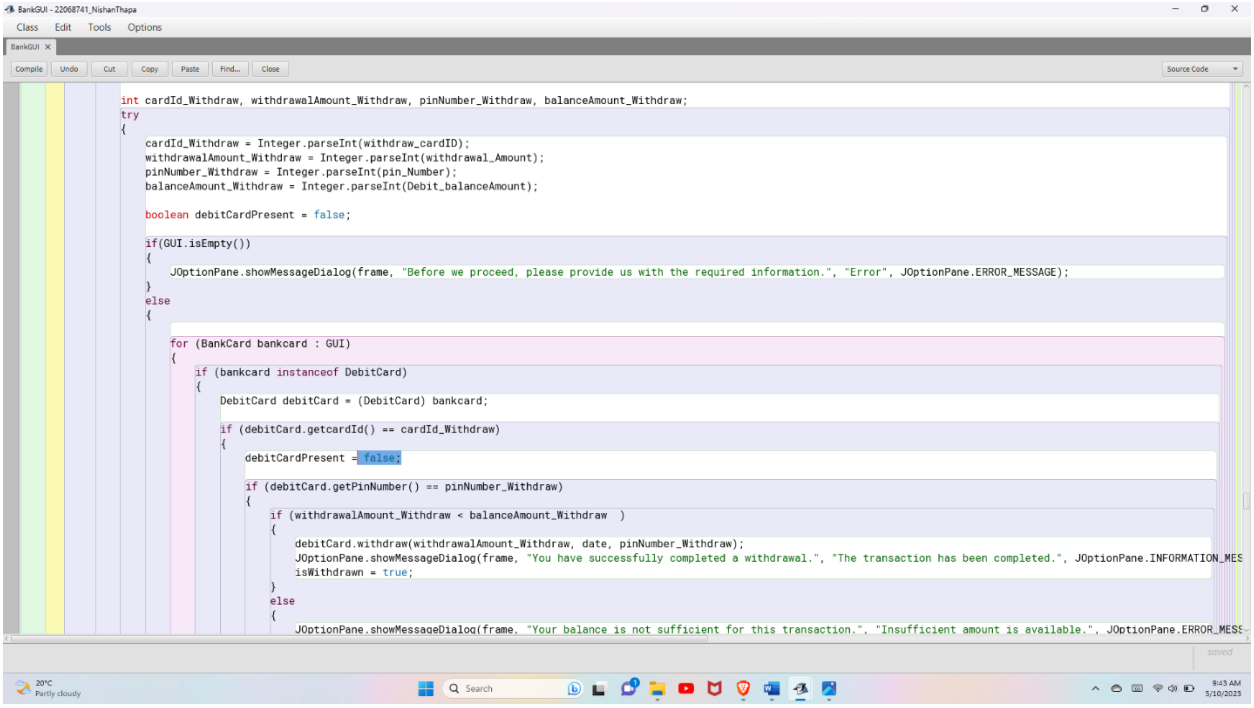

Figure 35 screenshot of error 3

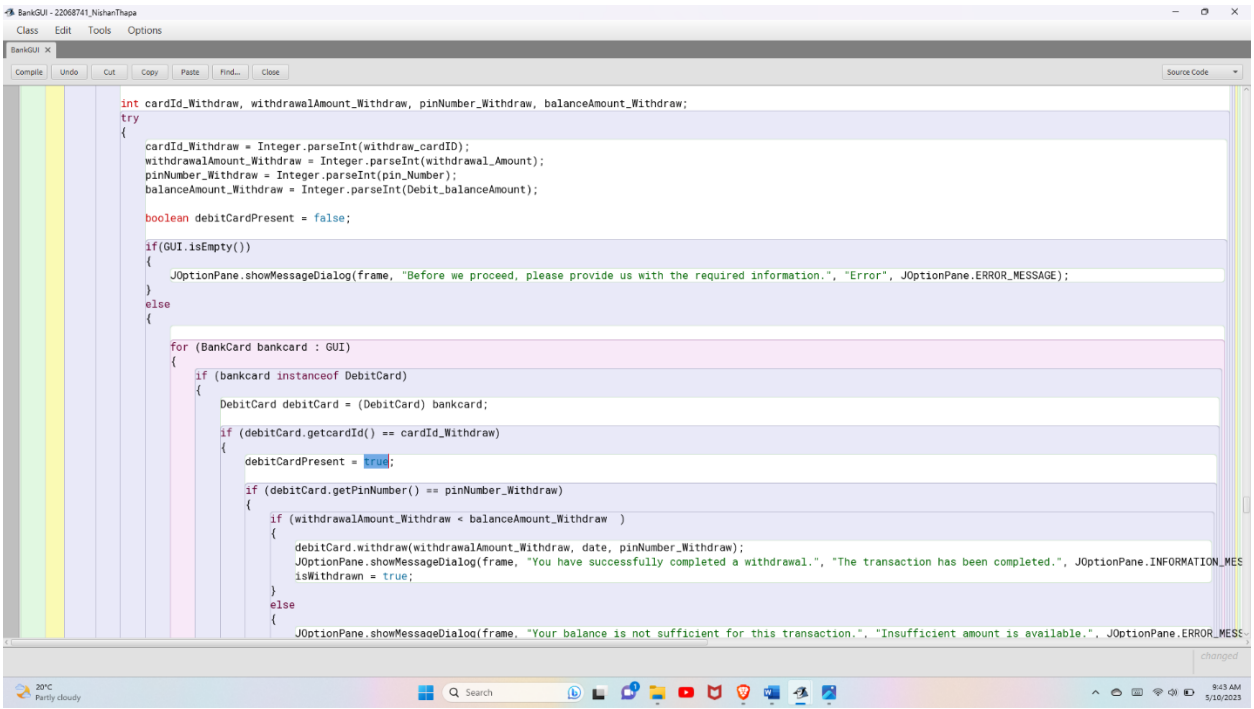

Figure 36 screenshot of error 3

9 Conclusion

Finally, the project required the development of a graphical user interface (GUI) for a private bank's debit and credit card operations. Users may utilize the GUI to accomplish a variety of tasks, such as withdrawing money from a debit card, setting a credit limit, and canceling a credit card. The usage of try-catch exception handling contributed to the program's stability and dependability.

We obtained a better understanding of GUI design concepts as well as the relevance of exception handling in designing robust and error-free systems as a result of this project. However, we encountered certain challenges in implementing specific features, such as credit limit setting, logic behind the add buttons of the credit and debit card and the withdrawal button for debit card which made me realize the importance of logic required for better programming skills.

Overall, this project gave us a wonderful opportunity to exercise and use our Java programming skills in a real-world setting, as well as to improve our problem-solving skills and software development techniques.

10 Appendix

```
/**
 * Write a description of class BankGUI here.
 *
 * @author (22068741 Nishan Thapa)
 * @version (1.0.0)
 */

//Step 0 : Import

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

```
import javax.swing.JPanel;

import java.util.ArrayList;

import javax.swing.JLabel;



public class BankGUI implements ActionListener  {

   //declare all components here

    private JFrame frame;

    private JTabbedPane panel;

   //Debit card components

    private JPanel debitCard_panel;

    private String debitCard_clientName;

    private String debitCard_issuerBank;
```

```
    private String debitCard_bankAccount;

    private int    debitCard_cardID;

    private int    debitCard_PIN_Number;

    private double debitCard_balanceAmount;




   //Debit card headings



    private JLabel debitCard;

    private JLabel add_debitCard;

    private JLabel debitCard_clientName_D;

    private JLabel debitCard_issuerBank_D;

    private JLabel debitCard_bankAccount_D;

    private JLabel debitCard_cardID_D;

    private JLabel debitCard_balanceAmount_D;

    private JLabel debitCard_PIN_Number_D;
```

```
//Debit card Textfields


 private JTextField debit_clientName;

 private JTextField debit_issuerBank;

 private JTextField debit_bankAccount;

 private JTextField debit_cardID;

 private JTextField debit_PIN_Number;

 private JTextField debit_balanceAmount;



//Debit card buttons components


 private JButton Add_debitCard;

 private JButton Display_debitCard;

 private JButton Clear_debitCard;


//Debit card withdrawal components
```

```
    private int debitWithdrawal_cardID;

    private int debitWithdrawal_amount;

    private int debitWithdrawal_PIN_Number;

    private String debit_DateOfWithdrawal;


   //Debit card withdrawal headings


    private JLabel withdraw_debitCard;

    private JLabel withdraw_amount;

    private JLabel withdraw_cardID;

    private JLabel withdraw_date;

    private JLabel withdraw_PIN_Number;


   //Debit card withdraw Textfields


    private JTextField withdraw_amount_T;
```

```
    private JTextField withdraw_cardID_T;

    private JTextField withdraw_PIN_Number_T;



   //Debit card withdrawl buttons components



    private JButton  withdrawal;

    private JButton  clearWithdrawal;

    private JButton  displayDebitCard;



   //Debit card combobox



    private JComboBox comboBox_Day;

    private JComboBox comboBox_Month;

    private JComboBox comboBox_Year;



   //------------Credit card components------------------------------
```

```
    private JPanel creditCard_panel;

    private String creditCard_clientName;

    private String creditCard_issuerBank;

    private String creditCard_bankAccount;

    private String creditCard_expirationDate;

    private int    creditCard_cardID;

    private int    cvc_Number;

    private double creditCard_balanceAmount;

    private double interestRate;



   //Credit card headings



    private JLabel creditCard;

    private JLabel Add_creditCard;

    private JLabel creditCard_clientName_C;

    private JLabel creditCard_cardID_C;

    private JLabel creditCard_issuerBank_C;
```

```
    private JLabel interestRate_C;

    private JLabel creditCard_bankAccount_C;

    private JLabel creditCard_balanceAmount_C;

    private JLabel cvc_Number_C;

    private JLabel creditCard_expirationDate_C;



   //Credit card TextFields



    private JTextField creditCard_clientName_T;

    private JTextField creditCard_cardID_T;

    private JTextField creditCard_issuerBank_T;

    private JTextField creditCard_interestRate_T;

    private JTextField creditCard_bankAccount_T;

    private JTextField creditCard_balanceAmount_T;

    private JTextField creditCard_CVCNumber_T;



    //Credit card combobox
```

```
    private JComboBox comboBox_Day_C;

    private JComboBox comboBox_Month_C;

    private JComboBox comboBox_Year_C;


    //Credit card buttons


    private JButton   add_creditCard_C;

    private JButton   display_creditCard;

    private JButton   clear_creditCard;


   //Credit limit components


    private int     limit_cardID;

    private boolean gracePeriod;

    private double  credit_limit;


   // Credit limit headings
```

```
    private JLabel creditCard_creditLimit;

    private JLabel creditCard_cardID_1;

    private JLabel creditCard_gracePeriod_1;

    private JLabel creditCard_creditLimit_1;


   //Credit limit TextFields


    private JTextField creditCard_cardID_t;

    private JTextField creditCard_gracePeriod_T;

    private JTextField creditCard_creditLimit_T;


   //Credit limit buttons


    private JButton set_creditLimit;

    private JButton clear_creditLimit;
```

```
//Cancel credit card components

 private String cancel_cardID;

//Cancel credit card headings

 private JLabel cancel_creditLimit_1;
 private JLabel cancel_cardID_1;

//Cancel credit card Textfield

 private JTextField cancel_cardID_T;

//Cancel credit card buttons

 private JButton cancel_creditLimit;
 private JButton cancel_creditCard;
```

```
// Arraylist


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




public BankGUI(){


//create a new frame


  //Step 1 : Create JFrame using constructor

  //Add components using contructor after Jframe and before displaying the Jframe

  //Set the bounds of the component (setBounds())

  //Add the component to the JFrame ()


JFrame frame = new JFrame();
```

```
//TabbedPane

JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.LEFT);

debitCard_panel = new JPanel();
//Step 3 : setting the layout managaer to null
debitCard_panel.setLayout(null);
tabbedPane.addTab("Debit Card", debitCard_panel);

//GUI Debit Card

//Debit card Labels

debitCard = new JLabel("Debit Card");
debitCard.setBounds(338, 29, 124, 36);
```

```
add_debitCard = new JLabel("Add Debit Card");

add_debitCard.setBounds(311, 96, 178, 36);



debitCard_clientName_D = new JLabel("Client Name: ");

debitCard_clientName_D.setBounds(35, 179, 77, 20);



debitCard_issuerBank_D = new JLabel("Issuer Bank: ");

debitCard_issuerBank_D.setBounds(35, 268, 84, 28);



debitCard_bankAccount_D = new JLabel("Bank Account: ");

debitCard_bankAccount_D.setBounds(35, 349, 102, 41);



debitCard_cardID_D = new JLabel("Card ID: ");

debitCard_cardID_D.setBounds(425, 179, 73, 33);



debitCard_PIN_Number_D = new JLabel("PIN Number: ");
```

```
debitCard_PIN_Number_D.setBounds(425, 274, 95, 25);



debitCard_balanceAmount_D = new JLabel("Balance Amount: ");

debitCard_balanceAmount_D.setBounds(420, 357, 102, 20);



//Debit card Textfields



debit_clientName = new JTextField();

debit_clientName.setBounds(151, 179, 180, 32);



debit_issuerBank = new JTextField();

debit_issuerBank.setBounds(151, 268, 180, 32);



debit_bankAccount = new JTextField();

debit_bankAccount.setBounds(151, 358, 180, 32);



debit_cardID = new JTextField();
```

```
debit_cardID.setBounds(559, 179, 180, 32);



debit_PIN_Number = new JTextField();

debit_PIN_Number.setBounds(559, 271, 180, 32);



debit_balanceAmount = new JTextField();

debit_balanceAmount.setBounds(559, 357, 180, 32);



//Debit card Buttons



Add_debitCard = new JButton("Add Debit Card");

Add_debitCard.setBounds(131, 442, 120, 32);

Add_debitCard.addActionListener(this);



Display_debitCard = new JButton("Display");

Display_debitCard.setBounds(339, 443, 120, 32);

Display_debitCard.addActionListener(this);
```

```
Clear_debitCard = new JButton("Clear");

Clear_debitCard.setBounds(547, 443, 120, 32);

Clear_debitCard.addActionListener(this);


// Debit card Withdrawal headings


withdraw_debitCard = new JLabel("Withdraw from Debit Card ");

withdraw_debitCard.setBounds(252, 526, 319, 36);


withdraw_amount = new JLabel("Amount of Withdrawal: ");

withdraw_amount.setBounds(35, 604, 137, 20);


withdraw_cardID = new JLabel("Card ID: ");

withdraw_cardID.setBounds(35, 676, 48, 20);


withdraw_date = new JLabel("Date of Withdrawal: ");
```

```
withdraw_date.setBounds(428, 603, 121, 20);



withdraw_PIN_Number = new JLabel("PIN Number: ");

withdraw_PIN_Number.setBounds(428, 671, 77, 20);



//Debit Card Withdrawl TextFields



withdraw_amount_T = new JTextField();

withdraw_amount_T.setBounds(190, 604, 180, 32);



withdraw_cardID_T = new JTextField();

withdraw_cardID_T.setBounds(190, 676, 180, 32);



withdraw_PIN_Number_T = new JTextField();

withdraw_PIN_Number_T.setBounds(580, 671, 180, 32);



//Debit card ComboBox
```

```
comboBox_Day = new JComboBox();
comboBox_Day.addItem("1");
comboBox_Day.addItem("2");
comboBox_Day.addItem("3");
comboBox_Day.addItem("4");
comboBox_Day.addItem("5");
comboBox_Day.addItem("6");
comboBox_Day.addItem("7");
comboBox_Day.addItem("8");
comboBox_Day.addItem("9");
comboBox_Day.addItem("10");
comboBox_Day.addItem("11");
comboBox_Day.addItem("12");
comboBox_Day.addItem("13");
comboBox_Day.addItem("14");
comboBox_Day.addItem("15");
```

```
comboBox_Day.addItem("16");
comboBox_Day.addItem("17");
comboBox_Day.addItem("18");
comboBox_Day.addItem("19");
comboBox_Day.addItem("20");
comboBox_Day.addItem("21");
comboBox_Day.addItem("22");
comboBox_Day.addItem("23");
comboBox_Day.addItem("24");
comboBox_Day.addItem("25");
comboBox_Day.addItem("26");
comboBox_Day.addItem("27");
comboBox_Day.addItem("28");
comboBox_Day.addItem("29");
comboBox_Day.addItem("30");
comboBox_Day.addItem("31");
comboBox_Day.setBounds(580, 603, 50, 34);
```

```
comboBox_Month = new JComboBox();
comboBox_Month.addItem("1");
comboBox_Month.addItem("2");
comboBox_Month.addItem("3");
comboBox_Month.addItem("4");
comboBox_Month.addItem("5");
comboBox_Month.addItem("6");
comboBox_Month.addItem("7");
comboBox_Month.addItem("8");
comboBox_Month.addItem("9");
comboBox_Month.addItem("10");
comboBox_Month.addItem("11");
comboBox_Month.addItem("12");
comboBox_Month.setBounds(649, 603, 51, 33);

comboBox_Year = new JComboBox();
```

```
    comboBox_Year.addItem("2010");

    comboBox_Year.addItem("2011");

    comboBox_Year.addItem("2012");

    comboBox_Year.addItem("2013");

    comboBox_Year.addItem("2014");

    comboBox_Year.addItem("2015");

    comboBox_Year.addItem("2016");

    comboBox_Year.addItem("2017");

    comboBox_Year.addItem("2018");

    comboBox_Year.addItem("2019");

    comboBox_Year.addItem("2020");

    comboBox_Year.addItem("2021");

    comboBox_Year.addItem("2022");

    comboBox_Year.addItem("2023");

    comboBox_Year.setBounds(719, 603, 47, 33);



    //Debit card Withdrawal buttons
```

```
withdrawal = new JButton("Withdraw");

withdrawal.setBounds(144, 746, 120, 32);

withdrawal.addActionListener(this);


displayDebitCard = new JButton("Display");

displayDebitCard.setBounds(351, 746, 120, 32);

displayDebitCard.addActionListener(this);


clearWithdrawal = new JButton("Clear");

clearWithdrawal.setBounds(545, 746, 120, 32);

clearWithdrawal.addActionListener(this);


//Adding items to debit card


debitCard_panel.add(debitCard);

debitCard_panel.add(add_debitCard);
```

```
debitCard_panel.add(debitCard_clientName_D);

debitCard_panel.add(debitCard_issuerBank_D);

debitCard_panel.add(debitCard_bankAccount_D);

debitCard_panel.add(debitCard_cardID_D);

debitCard_panel.add(debitCard_PIN_Number_D);

debitCard_panel.add(debitCard_balanceAmount_D);

debitCard_panel.add(debit_clientName);

debitCard_panel.add(debit_issuerBank);

debitCard_panel.add(debit_bankAccount);

debitCard_panel.add(debit_cardID);

debitCard_panel.add(debit_PIN_Number);

debitCard_panel.add(debit_balanceAmount);

debitCard_panel.add(Add_debitCard);

debitCard_panel.add(Display_debitCard);

debitCard_panel.add(Clear_debitCard);

debitCard_panel.add(withdraw_debitCard);

debitCard_panel.add(withdraw_amount);
```

```
debitCard_panel.add(withdraw_cardID);

debitCard_panel.add(withdraw_date);

debitCard_panel.add(withdraw_PIN_Number);

debitCard_panel.add(withdraw_amount_T);

debitCard_panel.add(withdraw_cardID_T);

debitCard_panel.add(withdraw_PIN_Number_T);

debitCard_panel.add(comboBox_Day);

debitCard_panel.add(comboBox_Month);

debitCard_panel.add(comboBox_Year);

debitCard_panel.add(withdrawal);

debitCard_panel.add(displayDebitCard);

debitCard_panel.add(clearWithdrawal);



//----------------- Credit card panel-----------------------------------------------------------



creditCard_panel = new JPanel();

//Step 3 : setting the layout managaer to null
```

```
creditCard_panel.setLayout(null);

tabbedPane.addTab("Credit Card", creditCard_panel);



//Credit card heading



creditCard = new JLabel("Credit Card");

creditCard.setBounds(333, 19, 133, 36);



//Add Credit card heading




Add_creditCard = new JLabel("Add Credit Card");

Add_creditCard.setBounds(301, 69, 186, 36);



//Add Credit card components heading



creditCard_clientName_C = new JLabel("Client Name: ");
```

```
creditCard_clientName_C.setBounds(28, 136, 77, 49);


creditCard_cardID_C = new JLabel("Card ID: ");

creditCard_cardID_C.setBounds(28, 213, 48, 20);


creditCard_issuerBank_C = new JLabel("Issuer Bank: ");

creditCard_issuerBank_C.setBounds(28, 281, 74, 20);


interestRate_C = new JLabel("Interest Rate: ");

interestRate_C.setBounds(28, 349, 80, 20);


creditCard_bankAccount_C = new JLabel("Bank Account: ");

creditCard_bankAccount_C.setBounds(415, 135, 86, 46);


creditCard_balanceAmount_C = new JLabel("Balance Amount: ");

creditCard_balanceAmount_C.setBounds(415, 211, 102, 22);
```

```
cvc_Number_C = new JLabel("CVC Number: ");

cvc_Number_C.setBounds(415, 281, 108, 22);



creditCard_expirationDate_C = new JLabel("Expiration Date: ");

creditCard_expirationDate_C.setBounds(415, 351, 96, 20);



//Credit card components TextFields



creditCard_clientName_T = new JTextField();

creditCard_clientName_T.setBounds(127, 145, 180, 32);



creditCard_cardID_T = new JTextField();

creditCard_cardID_T.setBounds(127, 213, 180, 32);



creditCard_issuerBank_T = new JTextField();

creditCard_issuerBank_T.setBounds(127, 281, 180, 32);
```

```
creditCard_interestRate_T = new JTextField();

creditCard_interestRate_T.setBounds(127, 349, 180, 32);



creditCard_bankAccount_T = new JTextField();

creditCard_bankAccount_T.setBounds(551, 141, 180, 32);



creditCard_balanceAmount_T = new JTextField();

creditCard_balanceAmount_T.setBounds(551, 211, 180, 32);



creditCard_CVCNumber_T = new JTextField();

creditCard_CVCNumber_T.setBounds(551, 281, 180, 32);



//Credit card components comboBox



comboBox_Day_C = new JComboBox();

comboBox_Day_C.addItem("1");

comboBox_Day_C.addItem("2");
```

```
comboBox_Day_C.addItem("3");

comboBox_Day_C.addItem("4");

comboBox_Day_C.addItem("5");

comboBox_Day_C.addItem("6");

comboBox_Day_C.addItem("7");

comboBox_Day_C.addItem("8");

comboBox_Day_C.addItem("9");

comboBox_Day_C.addItem("10");

comboBox_Day_C.addItem("11");

comboBox_Day_C.addItem("12");

comboBox_Day_C.addItem("13");

comboBox_Day_C.addItem("14");

comboBox_Day_C.addItem("15");

comboBox_Day_C.addItem("16");

comboBox_Day_C.addItem("17");

comboBox_Day_C.addItem("18");

comboBox_Day_C.addItem("19");
```

```
comboBox_Day_C.addItem("20");

comboBox_Day_C.addItem("21");

comboBox_Day_C.addItem("22");

comboBox_Day_C.addItem("23");

comboBox_Day_C.addItem("24");

comboBox_Day_C.addItem("25");

comboBox_Day_C.addItem("26");

comboBox_Day_C.addItem("27");

comboBox_Day_C.addItem("28");

comboBox_Day_C.addItem("29");

comboBox_Day_C.addItem("30");

comboBox_Day_C.addItem("31");

comboBox_Day_C.setBounds(551, 359, 50, 19);



comboBox_Month_C = new JComboBox();

comboBox_Month_C.addItem("1");

comboBox_Month_C.addItem("2");
```

```
comboBox_Month_C.addItem("3");

comboBox_Month_C.addItem("4");

comboBox_Month_C.addItem("5");

comboBox_Month_C.addItem("6");

comboBox_Month_C.addItem("7");

comboBox_Month_C.addItem("8");

comboBox_Month_C.addItem("9");

comboBox_Month_C.addItem("10");

comboBox_Month_C.addItem("11");

comboBox_Month_C.addItem("12");

comboBox_Month_C.setBounds(619, 359, 50, 19);



comboBox_Year_C = new JComboBox();

comboBox_Year_C.addItem("2010");

comboBox_Year_C.addItem("2011");

comboBox_Year_C.addItem("2012");

comboBox_Year_C.addItem("2013");
```

```
comboBox_Year_C.addItem("2014");

comboBox_Year_C.addItem("2015");

comboBox_Year_C.addItem("2016");

comboBox_Year_C.addItem("2017");

comboBox_Year_C.addItem("2018");

comboBox_Year_C.addItem("2019");

comboBox_Year_C.addItem("2020");

comboBox_Year_C.addItem("2021");

comboBox_Year_C.addItem("2022");

comboBox_Year_C.addItem("2023");

comboBox_Year_C.setBounds(687, 359, 50, 19);



//ADD Credit Buttons



add_creditCard_C = new JButton("Add Credit Card");

add_creditCard_C.setBounds(79, 431, 148, 32);

add_creditCard_C.addActionListener(this);
```

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

display_creditCard.setBounds(324, 431, 120, 32);

display_creditCard.addActionListener(this);


clear_creditCard = new JButton("Clear");

clear_creditCard.setBounds(541, 431, 120, 32);

clear_creditCard.addActionListener(this);


//Set credit limit heading


creditCard_creditLimit = new JLabel("Set Credit Limit");

creditCard_creditLimit.setBounds(311, 506, 178, 36);


//Set credit limit components labels


creditCard_cardID_1 = new JLabel("Card ID");
```

```
creditCard_cardID_1.setBounds(31, 564, 48, 17);



creditCard_gracePeriod_1 = new JLabel("Grace Period");

creditCard_gracePeriod_1.setBounds(31, 635, 82, 20);



creditCard_creditLimit_1 = new JLabel("Credit Limit");

creditCard_creditLimit_1.setBounds(444, 561, 75, 20);



//Set Credit components TextFields



creditCard_cardID_t = new JTextField();

creditCard_cardID_t.setBounds(179, 561, 180, 32);



creditCard_gracePeriod_T = new JTextField();

creditCard_gracePeriod_T.setBounds(179, 635, 180, 32);



creditCard_creditLimit_T = new JTextField();
```

```
creditCard_creditLimit_T.setBounds(562, 561, 180, 32);



//Set Credit Buttons



set_creditLimit = new JButton("Set Credit Limit");

set_creditLimit.setBounds(450, 629, 130, 32);

set_creditLimit.addActionListener(this);



clear_creditLimit = new JButton("Clear");

clear_creditLimit.setBounds(622, 629, 120, 32);

clear_creditLimit.addActionListener(this);



//Cancel credit components heading



cancel_creditLimit_1 = new JLabel("Cancel Credit Card");

cancel_creditLimit_1.setBounds(311, 689, 221, 36);
```

```
//Cancel credit Labels



cancel_cardID_1 = new JLabel("Card ID");

cancel_cardID_1.setBounds(31, 752, 48, 17);



//Cancel credit TextField



cancel_cardID_T = new JTextField();

cancel_cardID_T.setBounds(121, 752, 180, 32);



//Cancel credit buttons



cancel_creditLimit = new JButton("Cancel Credit Card");

cancel_creditLimit.setBounds(411, 753, 159, 32);

cancel_creditLimit.addActionListener(this);



cancel_creditCard = new JButton("Clear");
```

```
cancel_creditCard.setBounds(627, 753, 120, 32);

cancel_creditCard.addActionListener(this);



//Adding components to credit panel



creditCard_panel.add(creditCard);

creditCard_panel.add(Add_creditCard);

creditCard_panel.add(creditCard_clientName_C);

creditCard_panel.add(creditCard_cardID_C);

creditCard_panel.add(creditCard_issuerBank_C);

creditCard_panel.add(interestRate_C);

creditCard_panel.add(creditCard_bankAccount_C);

creditCard_panel.add(creditCard_balanceAmount_C);

creditCard_panel.add(cvc_Number_C);

creditCard_panel.add(creditCard_expirationDate_C);

creditCard_panel.add(creditCard_clientName_T);

creditCard_panel.add(creditCard_cardID_T);
```

```
creditCard_panel.add(creditCard_issuerBank_T);

creditCard_panel.add(creditCard_interestRate_T);

creditCard_panel.add(creditCard_bankAccount_T);

creditCard_panel.add(creditCard_balanceAmount_T);

creditCard_panel.add(creditCard_CVCNumber_T);

creditCard_panel.add(comboBox_Day_C);

creditCard_panel.add(comboBox_Month_C);

creditCard_panel.add(comboBox_Year_C);

creditCard_panel.add(add_creditCard_C);

creditCard_panel.add(display_creditCard);

creditCard_panel.add(clear_creditCard);

creditCard_panel.add(creditCard_creditLimit);

creditCard_panel.add(creditCard_cardID_1);

creditCard_panel.add(creditCard_gracePeriod_1);

creditCard_panel.add(creditCard_creditLimit_1 );

creditCard_panel.add(creditCard_cardID_t);

creditCard_panel.add(creditCard_gracePeriod_T);
```

```
creditCard_panel.add(creditCard_creditLimit_T);

creditCard_panel.add(set_creditLimit);

creditCard_panel.add(clear_creditLimit);

creditCard_panel.add(cancel_creditLimit_1);

creditCard_panel.add(cancel_cardID_1);

creditCard_panel.add(cancel_cardID_T);

creditCard_panel.add(cancel_creditLimit);

creditCard_panel.add(cancel_creditCard);




frame.add(tabbedPane);



//Step 2 : set the size of the Jframe

frame.setSize(800,800);

//Step 4 : set what happens when the frame is closed(default close)

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//Step 5 : make the frame visible
```

```
    frame.setVisible(true);



    }



    @Override



    public void actionPerformed(ActionEvent e){



      //Add button for Debit card



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

          //if arrayList is empty

          //Add Debit Card

          if(debit_clientName.getText().isEmpty()     ||     debit_issuerBank.getText().isEmpty()     ||
debit_bankAccount.getText().isEmpty()          ||          debit_cardID.getText().isEmpty()          ||
debit_PIN_Number.getText().isEmpty() || debit_balanceAmount.getText().isEmpty()){

               JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out
the form that has been given to you.","Warning",JOptionPane.WARNING_MESSAGE);
```

```
    }
      //add logic
      //add debit card
        //get the parameters of the constructor
        //call the constructor | create the object
      //add to array list
      //show message:("Card has been added")


      //else
            //check to see if cardId is present
              //loop through the arrayList
              //if present:
                //dont add
                //show message ("CAnt be added same ID")
               //else:
             //Add Debit Card
             //show message ("Debit card has been added")
```

```
        else{
          try{
            double debitCard_balanceAmount =
Integer.parseInt(debit_balanceAmount.getText());
            int debitCard_cardID = Integer.parseInt(debit_cardID.getText());
            int debitCard_PIN_Number = Integer.parseInt(debit_PIN_Number.getText());
            String debitCard_clientName = debit_clientName.getText();
            String debitCard_bankAccount = debit_bankAccount.getText();
            String debitCard_issuerBank = debit_issuerBank.getText();
            if(GUI.isEmpty()){
              DebitCard Debit_Parameters = new
DebitCard(debitCard_balanceAmount,debitCard_cardID,debitCard_bankAccount,debitCard_issu
erBank,debitCard_clientName,debitCard_PIN_Number);
              GUI.add(Debit_Parameters);

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

```
                    DebitCard debitCard = (DebitCard)card;
                    if(debitCard.getcardId() == debitCard_cardID){
                      JOptionPane.showMessageDialog(frame,"The addition of a Debit Card to the account has been confirmed.");
                    }else{
                      DebitCard Debit_Parameters = new DebitCard(debitCard_balanceAmount,debitCard_cardID,debitCard_bankAccount,debitCard_issuerBank,debitCard_clientName,debitCard_PIN_Number);
                      GUI.add(Debit_Parameters);
                      JOptionPane.showMessageDialog(frame,"The addition of the Debit Card was successful.");
                    }
                  }
                }
              }

                    //get the parameters
                //check if the parameter are valid or not
```

```
                    //if not valid:

                      //show message:



            }catch(Exception ex){

              JOptionPane.showMessageDialog(frame, "Kindly review the provided form and ensure that all information has been accurately provided.", "Warning", JOptionPane.WARNING_MESSAGE);

            }

          }

      }



      if (debit_PIN_Number.getText().length()>4){

          JOptionPane.showMessageDialog(frame,"Your PIN Number input should not exceed four digits. Please try again.","Error",JOptionPane.WARNING_MESSAGE);

       }



      //display buton functionality for debitcard
```

```
    //call the display method
    if(e.getSource() == Display_debitCard){
        //if arraylist is empty:
         // cant display
         // show message: Cant display
        if(debit_clientName.getText().isEmpty()  ||  debit_issuerBank.getText().isEmpty()  ||
debit_bankAccount.getText().isEmpty()  ||  debit_cardID.getText().isEmpty()  ||
debit_PIN_Number.getText().isEmpty() || debit_balanceAmount.getText().isEmpty()){
            JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out
the form that has been given to you.","Warning",JOptionPane.WARNING_MESSAGE);
        }


        //else:
         // loop through the arraylist
         // if debit card is present(if obj instance of DebitCard)
            // downcast
            // call the display method
        else{
```

```
            try{



            }catch(Exception ex){

                JOptionPane.showMessageDialog(frame,"The action could not be completed due to
an exception. Please try again.","Error",JOptionPane.WARNING_MESSAGE);

            }

        }

        for(BankCard Debit: GUI  ){

            if( Debit instanceof DebitCard){

                ((DebitCard)Debit).display();

            }

        }

    }



    //Clear button function for Add Debit card



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

        debit_clientName.setText("");
```

```
        debit_issuerBank.setText("");

        debit_bankAccount.setText("");

        debit_cardID.setText("");

        debit_PIN_Number.setText("");

        debit_balanceAmount.setText("");

    }



    //Withdrawal button for debit card

        //get the parameters

        //call the withdraw method



        //if arraylist is empty

            //dont withdraw

            //show message

    if (e.getSource() == withdrawal)

     {

         String withdraw_cardID     = withdraw_cardID_T.getText();
```

```
        String withdrawal_Amount    = withdraw_amount_T.getText();

        String Debit_balanceAmount = debit_balanceAmount.getText();

        String pin_Number          = withdraw_PIN_Number_T.getText();

        String Year                = comboBox_Year.getSelectedItem().toString();

        String Month               = comboBox_Month.getSelectedItem().toString();

        String Day                 = comboBox_Day.getSelectedItem().toString();

        String date                = Day +"/"+ Month  + "/" +Year;

        boolean isWithdrawn        = false;



        if (withdraw_cardID_T.getText().isEmpty() || withdraw_amount_T.getText().isEmpty() ||
withdraw_PIN_Number_T.getText().isEmpty())

        {

           JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out
the form that has been given to you.","Warning", JOptionPane.ERROR_MESSAGE);

        }



        //else

        //check if the debitcard is present
```

```
            //loop through the arrayList
            //is is debitcard or not?
            //if card id matches
              //show information in a dialogue box
              //if pin number matches
                //call the withdraw method with the parameters
                //show message: Amoount has been withdrawn


        else
        {


            int cardId_Withdraw, withdrawalAmount_Withdraw, pinNumber_Withdraw,
balanceAmount_Withdraw;
            try
            {
               cardId_Withdraw = Integer.parseInt(withdraw_cardID);
               withdrawalAmount_Withdraw = Integer.parseInt(withdrawal_Amount);
               pinNumber_Withdraw = Integer.parseInt(pin_Number);
```

```
balanceAmount_Withdraw = Integer.parseInt(Debit_balanceAmount);



boolean debitCardPresent = false;



if(GUI.isEmpty())

{

   JOptionPane.showMessageDialog(frame, "Before we proceed, please provide us with the required information.", "Error", JOptionPane.ERROR_MESSAGE);

}

else

{



   for (BankCard bankcard : GUI)

   {

      if (bankcard instanceof DebitCard)

      {

         DebitCard debitCard = (DebitCard) bankcard;
```

```
                if (debitCard.getcardId() == cardId_Withdraw)
                {
                   debitCardPresent = true;


                   if (debitCard.getPinNumber() == pinNumber_Withdraw)
                   {
                      if (withdrawalAmount_Withdraw < balanceAmount_Withdraw  )
                      {
                         debitCard.withdraw(withdrawalAmount_Withdraw, date,
pinNumber_Withdraw);
                         JOptionPane.showMessageDialog(frame, "You have successfully
completed a withdrawal.", "The transaction has been completed.",
JOptionPane.INFORMATION_MESSAGE);
                         isWithdrawn = true;
                      }
                      else
                      {
```

```
                        JOptionPane.showMessageDialog(frame, "Your balance is not sufficient
for this transaction.", "Insufficient amount is available.", JOptionPane.ERROR_MESSAGE);
                        }
                    }else
                    {
                        JOptionPane.showMessageDialog(frame, "The pin number you entered
does not match our records.", "The pin number entered is incorrect.",
JOptionPane.ERROR_MESSAGE);
                    }

                    break;
                }
            }

        }

        //else
        //show message card not found
```

```
                if (!debitCardPresent)
                {
                    JOptionPane.showMessageDialog(frame, "The provided ID does not match any debit cards in our system.", "The system was unable to find the card.", JOptionPane.ERROR_MESSAGE);
                }
            }
        }
        catch(NumberFormatException n)
        {
            JOptionPane.showMessageDialog(frame, "Please enter digits only, other characters are not accepted.", "The input provided is incorrect.", JOptionPane.ERROR_MESSAGE);
        }
    }
}

if (withdraw_PIN_Number_T.getText().length()>4){
```

```
        JOptionPane.showMessageDialog(frame,"Your PIN Number input should not exceed four digits. Please try again.","Error",JOptionPane.WARNING_MESSAGE);
      }


      //display buton for withdraw debit card
     if(e.getSource() == displayDebitCard){


        if(withdraw_amount_T.getText().isEmpty()  ||  withdraw_cardID_T.getText().isEmpty()  ||  withdraw_PIN_Number_T.getText().isEmpty()){
            JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out the form that has been given to you.","Warning",JOptionPane.WARNING_MESSAGE);


        }else{
            try{


            }catch(Exception ex){
                JOptionPane.showMessageDialog(frame,"The action could not be completed due to an exception. Please try again.","Error",JOptionPane.WARNING_MESSAGE);
```

```
            }
        }
        for(BankCard Debit: GUI  ){
            if( Debit instanceof DebitCard){
                ((DebitCard)Debit).display();
            }
        }
    }

    //Clear button function for Withdrawal Debit card

    if(e.getSource() == clearWithdrawal){
        withdraw_amount_T.setText("");
        withdraw_cardID_T.setText("");
        withdraw_PIN_Number_T.setText("");
    }
```

```
//Same logic as the debit card
if(e.getSource() == add_creditCard_C){
    if(creditCard_clientName_T.getText().isEmpty() || creditCard_cardID_T.getText().isEmpty() || creditCard_issuerBank_T.getText().isEmpty() || creditCard_interestRate_T.getText().isEmpty() || creditCard_bankAccount_T.getText().isEmpty() || creditCard_balanceAmount_T.getText().isEmpty() || creditCard_CVCNumber_T.getText().isEmpty()){
        JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out the form that has been given to you.","Warning",JOptionPane.WARNING_MESSAGE);
    }
    else{
      try{
        String creditCard_clientName  = creditCard_clientName_T.getText();
        String creditCard_issuerBank  = creditCard_issuerBank_T.getText();
        String creditCard_bankAccount  = creditCard_bankAccount_T.getText();
        int  creditCard_cardID  = Integer.parseInt(creditCard_cardID_T.getText());
        int cvc_Number  = Integer.parseInt(creditCard_CVCNumber_T.getText());
```

```
            double creditCard_balanceAmount =
Integer.parseInt(creditCard_balanceAmount_T.getText());
            double interestRate = Integer.parseInt(creditCard_interestRate_T.getText());
            String year = comboBox_Year_C.getSelectedItem().toString();
            String month = comboBox_Month_C.getSelectedItem().toString();
            String day = comboBox_Day_C.getSelectedItem().toString();
            String creditCard_expirationDate = day +"/"+ month + "/" + year;
            if(GUI.isEmpty()){
              CreditCard Credit_Parameters = new CreditCard(creditCard_cardID
,creditCard_clientName,creditCard_issuerBank
,creditCard_bankAccount,creditCard_balanceAmount,cvc_Number
,interestRate,creditCard_expirationDate);
              GUI.add(Credit_Parameters);
            }else{
              for(BankCard card : GUI){
                if(card instanceof CreditCard){
                  CreditCard creditCard = (CreditCard)card;
                  if(creditCard.getcardId() == creditCard_cardID ){
```

```
                    JOptionPane.showMessageDialog(frame,"The addition of a Credit Card to
the account has been confirmed.");
                }else{
                    CreditCard Credit_Parameters = new CreditCard(creditCard_cardID
,creditCard_clientName,creditCard_issuerBank
,creditCard_bankAccount,creditCard_balanceAmount,cvc_Number
,interestRate,creditCard_expirationDate);
                    GUI.add(Credit_Parameters);
                    JOptionPane.showMessageDialog(frame,"The addition of the Credit Card
was successful.");
                }
              }
            }
          }
        }catch(Exception ex){
          JOptionPane.showMessageDialog(frame, "Kindly review the provided form and
ensure that all information has been accurately provided.", "Warning",
JOptionPane.WARNING_MESSAGE);
        }
```

```
        }
    }

    //display buton for creditcard

    if(e.getSource() == display_creditCard){
        if(creditCard_clientName_T.getText().isEmpty() || creditCard_cardID_T.getText().isEmpty() || creditCard_issuerBank_T.getText().isEmpty() || creditCard_interestRate_T.getText().isEmpty() || creditCard_bankAccount_T.getText().isEmpty() || creditCard_balanceAmount_T.getText().isEmpty() || creditCard_CVCNumber_T.getText().isEmpty()){
            JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out the form that has been given to you.","Warning",JOptionPane.WARNING_MESSAGE);
        }
        else{
            try{

            }catch(Exception ex){
```

```
                JOptionPane.showMessageDialog(frame,"The action could not be completed due to an exception. Please try again.","Error",JOptionPane.WARNING_MESSAGE);
            }
        }
        for(BankCard Credit : GUI  ){
            if( Credit instanceof CreditCard){
                ((CreditCard)Credit).display();
            }
        }
    }

    //Clear button function for Clear Credit card

    if(e.getSource() == clear_creditCard){
        creditCard_clientName_T.setText("");
        creditCard_cardID_T.setText("");
        creditCard_issuerBank_T.setText("");
        creditCard_interestRate_T.setText("");
```

```
        creditCard_bankAccount_T.setText("");

        creditCard_balanceAmount_T.setText("");

        creditCard_CVCNumber_T.setText("");

    }


    //Set Credit limit button functionality


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

        String creditLimit  =  creditCard_creditLimit_T.getText();

        String gracePeriod  =  creditCard_gracePeriod_T.getText();

        String limit_cardID =  creditCard_cardID_t.getText();


        if (limit_cardID.isEmpty() || gracePeriod.isEmpty()|| creditLimit.isEmpty())  {

            JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out
the form that has been given to you","Warning",JOptionPane.WARNING_MESSAGE);

        } else {

            try{

                int Creditlimit_T = Integer.parseInt(creditLimit);
```

```
int gracePeriod_T = Integer.parseInt(gracePeriod);

int limit_cardID_T = Integer.parseInt(limit_cardID);



boolean creditCardFound = false;



if(GUI.isEmpty()){

   JOptionPane.showMessageDialog(frame, "Before we proceed, please provide us with the required information.", "Error", JOptionPane.ERROR_MESSAGE);

}

else{



}



for (BankCard bankCard : GUI) {



   if (bankCard instanceof CreditCard) {

      CreditCard creditCard = (CreditCard) bankCard;
```

```
                if (creditCard.getcardId() == limit_cardID_T){
                    creditCardFound = true;

                    creditCard.setCreditLimit(Creditlimit_T, gracePeriod_T);
                    JOptionPane.showMessageDialog(frame, "The new credit limit has been added to your account.", "Successful", JOptionPane.INFORMATION_MESSAGE);
                    break;
                }
            }
        }

        if (!creditCardFound) {
            JOptionPane.showMessageDialog(frame, "The credit card with the provided ID could not be found.", " not found", JOptionPane.ERROR_MESSAGE);
        }

    }catch(NumberFormatException n){
```

```
                JOptionPane.showMessageDialog(frame, "Please enter digits only, other characters
are not accepted.", " Error", JOptionPane.ERROR_MESSAGE);
            }



        }
    }

    //Clear button function for Credit Limit


     if(e.getSource() == clear_creditLimit){
        creditCard_cardID_t.setText("");
        creditCard_gracePeriod_T.setText("");
        creditCard_creditLimit_T.setText("");
    }


    //Cancel credit button functionality


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

```
        String cardId_cancel = cancel_cardID_T.getText();



        try{
           if (cardId_cancel.isEmpty()){
              JOptionPane.showMessageDialog(frame, "Dear User, We kindly request that you fill out the form that has been given to you", "Error", JOptionPane.ERROR_MESSAGE);
           } else {
              int cardIdcancel = Integer.parseInt(cardId_cancel);
              boolean creditCardFound = false;
              for(BankCard bankcard : GUI) {
                 if(bankcard instanceof CreditCard) {
                    CreditCard creditcard = (CreditCard) bankcard;
                    if(creditcard.getcardId() == cardIdcancel) {
                       GUI.remove(creditcard);
                       JOptionPane.showMessageDialog(frame, "Credit Card " + cardIdcancel + "Id cancel", "Credit card Canceled", JOptionPane.INFORMATION_MESSAGE);
                       creditCardFound = true;
                       break;
```

```
                    }


                }
            }
            if (!creditCardFound) {
                JOptionPane.showMessageDialog(frame, "The credit card with the provided ID could not be found.", " not found", JOptionPane.ERROR_MESSAGE);
            }
        }


    }catch(NumberFormatException n){
        JOptionPane.showMessageDialog(frame, "Please enter digits only, other characters are not accepted.", "Error", JOptionPane.ERROR_MESSAGE);
    }
}


//Clear button function for Cancel Credit
```

```
    if(e.getSource() == cancel_creditCard){
      cancel_cardID_T.setText("");
    }



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

11 Bibliography

Bibliography

Experienceus, 2023. *What is wireframing | Experience UX.* [Online] Available at: https://www.experienceux.co.uk/faqs/what-is-wireframing/ [Accessed 26 January 2023].

Kim, G., 2020. *Balsamiq Review: Features, Pricing, Comparison - Bubble.* [Online] Available at: https://bubble.io/blog/balsamiq-review-bubble/ [Accessed 26 January 2023].

London, J. G. a. K. C., 2023. *BlueJ.* [Online] Available at: https://www.bluej.org/ [Accessed 10 may 2023].

S.R.L., S. E. S., 2023. *moqups.* [Online] Available at: https://moqups.com/ [Accessed 1 may 2023].

team, B. g. a. M., 2023. *Microsoft.* [Online] Available at: https://www.microsoft.com/en-us/?ql=2 [Accessed 10 may 2023].

University of York, 2023. *Tech basics: An introduction to text editors - University of York.* [Online] Available at: https://online.york.ac.uk/tech-basics-an-introduction-to-text-editors/ [Accessed 26 January 2023].