Module Code & Module Title:

CS4001NT Programming

Assessment Weightage & Type:

30% Individual Coursework

Year and Semester:

2022 Autumn

Student Name: Aayush Wanem Limbu

London Met ID: 22072043

College ID: np05cp4a220010

Assignment Due Date: 10th May 2023

Word Count: 11956

Contents

Table of Figures

Tables of Tables

1 Introduction

Java is a widely used object-oriented programming language and software platform runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many others. The rules and syntax of Java are based on the C and C++ languages.

One major advantage of developing software with Java is its portability. Once you have written code for a Java program on a notebook computer, it is very easy to move the code to a mobile device. When the language was invented in 1991 by James Gosling of Sun Microsystems (later acquired by Oracle), the primary goal was to be able to "write once, run anywhere."

The JVM, Java API, and a whole development environment make up the Java software platform. The Java bytecode is executed (parsed) by the JVM. The Extensible Markup Language (XML) generation, online services, networking, and security capabilities are only a few of the libraries that make up the Java API. The Java software platform and language when combined produce a potent, tested solution for corporate program development. Java is mainly used for developing mobile, desktop, web applications. It is also used in creating web servers and application servers.

Figure 1 Java Logo

1.1 Components of Java

Java has three different components for it to run and function. They are Java Development Kit (JDK)**,** Java Virtual Machine (JVM) and Java Runtime Environment (JRE). All three components are interconnected with each other and are needed to run a Java based program. Below is a short description about each of the components.

Java Development Kit (JDK)

JDK helps in building a development environment for applications or components based on Java programming Language. It helps in development**,** execution**,** debugging and documenting a program written with the Java Programming Language (Oracle, 2023).

Java Virtual Machine (JVM)

JVM can also be known as the foundation of Java Platform. It is an abstract machine which allows java program to run in any operating system which is also known as “write once, run anywhere” principle. It has instruction set in such a way that it manages and optimize program memory for the program execution (Tyson, 2022).

Java Runtime Environment (JRE)

JRE helps in communicating Java programming language with the operating system. It acts as translator between OS and the Java Program. JRE is made up of components like JVM Java class libraries and the Java class loaders. The JRE contains the libraries with prewritten codes which can be used when needed***** connecting with JVM**,** it executes the codes (Red Hat, 2020).

1.2 Bluej:

Blue J is a free, interactive Java development environment designed for beginners. It was Created by Michael Kölling and John Rosenberg at the University of Kent in Canterbury, England in 1999 and has since become a popular tool for teaching object-oriented programming (OOP) to students of all ages and experience levels.

BlueJ's main goal is to simplify the learning process by providing a visual representation of Java code. Its user interface resembles an integrated development environment (IDE) that students can use to write, test, and debug their Java programs. However, unlike other IDEs that are geared towards professional developers, BlueJ has a simplified interface that focuses on OOP concepts, making it an ideal tool for students and educators. One of the most unique features of BlueJ is its object bench, which provides a graphical representation of the objects Created by the program. This visual representation helps students understand how objects are Created stored, and manipulated in Java, which can be a challenging concept for beginners to grasp. Additionally, BlueJ provides a visual debugger that allows students to step through their code, see how it works, and identify any errors or bugs. BlueJ also includes features such as automatic indentation, syntax highlighting, and code completion, which help students write Java code with ease. It also supports a range of extensions and plug-ins that allow educators to customize the environment to meet their specific needs.

Overall, BlueJ is an excellent tool for anyone looking to learn or teach Java programming. Its simplified interface and visual representation of Java code make it an ideal learning environment for students of all ages and experience levels. Whether you're a beginner just starting to learn programming or an experienced developer looking to teach others, BlueJ is an excellent choice for your Java development needs.

Figure 2 Blue J logo image

1.3 Swing in Java:

Swing is a graphical user interface (GUI) toolkit for the Java programming language. It was introduced by Sun Microsystems (now Oracle Corporation) in 1997 as part of the Java Foundation Classes (JFC) and has since become a standard tool for creating desktop applications in Java.

Swing provides a SET of components and widgets that developers can use to CREATE graphical user interfaces, including buttons, text fields, menus, tables, and more. These components can be customized using various layout managers, which help to organize them on the screen and ensure they are properly aligned and sized.

One of the key benefits of Swing is that it is platform-independent, meaning that GUIs Created with Swing will look and function the same across different operating systems. This is because Swing uses a "look and feel" mechanism that allows it to adapt its appearance to match the underlying platform.

Swing also provides a range of advanced features, such as drag and drop support, data binding, and support for internationalization and accessibility. It also includes support for 2D and 3D graphics, allowing developers to CREATE sophisticated visual effects and animations.

Swing applications can be developed using any Java Integrated Development Environment (IDE), such as Eclipse, NetBeans, or IntelliJ IDEA. Additionally, Swing is fully compatible with other Java technologies, such as the Java Database Connectivity (JDBC) API and Java Servlets.

Despite its many benefits, Swing has faced some criticism in recent years due to the emergence of alternative GUI frameworks, such as JavaFX and SWT. However, Swing remains a popular and widely used tool for developing desktop applications in Java and its versatility and flexibility continue to make it a favorite among Java developers.

2 Class Diagram

The six different forms of structural diagrams include class diagrams. Class diagrams represent a system's static structure and are essential to the object modelling process. A single class diagram can be used to represent a whole system, or multiple class diagrams can represent the various parts of a system, depending on how complicated the system is. The blueprints for your system or subsystem are class diagrams. Class diagrams can be used to represent the system's components, show how they are related to one another, and explain the functions and services that each component performs. In many phases of system design, class diagrams are helpful. A class diagram might be useful during the analysis stage for understanding and identifying the components of your problem domain. In an early stage of an object-oriented software project, you draw class diagrams that contain classes that frequently transform into actual software classes and objects when you write code. Your prior analyses and conceptual models can later be improved into class diagrams that display the precise system components, user interfaces, logical implementations, and so on. Your class diagrams then take the form of a quick description of the operation of your system, the interactions between system components at various levels, and the implementation strategy for each component. Class diagrams can be used to represent, describe, and record structural elements in your models. For instance, you can design class diagrams to carry out the following tasks during the analysis and design phases of the development cycle:

- Capture and define the structure of classes and other classifiers.
- Define relationships between classes and classifiers.
- Illustrate the structure of a model by using attributes, operations, and signals.
- Show the common classifier roles and responsibilities that define the behavior of the system.
- Show the implementation classes in a package.
- Show the structure and behavior of one or more classes.
- Show an inheritance hierarchy among classes and classifiers.
- Show the workers and entities as business object models.

(IBM, 2021)

2.2 Class Diagram of Bank_GUI

Bank_GUI
- frame1, frame2, frame3: **JFrame** -cardId,clientName,issuerBank, bankAccount, balanceAmount, pinNumber, withdrawalAmount, withdrawalDate, CVCNumber, creditLimit, InterestRate, gracePeriod, expirationDate, withdrawcardId, withdrawpinNumber, creditcardId, creditClientName, creditIssuerBank, creditBankAccount, creditBalanceAmount : **JLabel** - textField1, textField2, textField3, textField4, textField5, textField6 textField7, textField8, textField9, textField10, textField11, textField12, textField13, textField14, textField15, textField16, textField17, textField18 : **JTextfield** -**ADD**debit, DisplayD, creditcard, clear, withdrawcard, withdraw, withdarwclear, withdrawback, back, addcredit, creditclear, addcreditLimit* cancelcredit, Displayc : **JButton** - years, months, days, years2, months2, days2 : **JComboBox<String>** -year[] : String -month[] : String -day[] :String INVALID : int array : ArrayList(BankCard)
+ BankGUI() + getCardId() : int

```
+ getClientName() : String
+ getBalanceAmount() : double
+ getPinNumber() : int
+ add Debit() : void
+ checkCardIdUnique(int cardId)  : boolean
+ showDebit() : void
+ getWCardId() : int
+ getWithdrawAmount()  : int
+ getwPinnumber() : int
+ getWithdrawalDate() : String
+ Withdraw() : void
+ showCredit() void
+ getCreditCardID() : int
+ getCreditClientName() : String
+ getCreditIssuerBank() : String
+  getCreditBankAccount() : String
+ getCreditBalanceAmount() : double
+ getCVCNumber() : int
+ getInterestRate() : double
+ getExpirationDate() : String
+ addCreditcard() : void
+ displayCredit() : void
+ getCardId() : int
+ getCreditLimit() : double
+ getGracePeriod() : int
+ creditLimit() : void
+ getCancelCardId() : int
+ cancelCreditCard() : void
+ main(String[] args) : void
```

Figure 3 figure of class diagram

2.3 Classes in Blue J interface

In BlueJ, a class is a basic building block for creating programs in Java. It's like a template or blueprint that defines the properties and behavior of objects that will be Created from it. To CREATE a class, you just need to right-click on the project view and select "New Class". You can ADD fields (variables) and methods (functions) to the class, and then CREATE objects from it by right-clicking on the class and selecting "New Object".

BlueJ also has a unique feature called the "object bench", which is a visual representation of the objects Created by the program. This helps you understand how the objects are Created and interact with each other. Additionally, BlueJ provides tools for testing and debugging your code, such as the debugger and JUnit integration.

Overall, understanding classes is essential for programming in BlueJ, and it's easy to create and use them to develop Java programs.

Shown below are the classes of Blue J interface of my coursework:

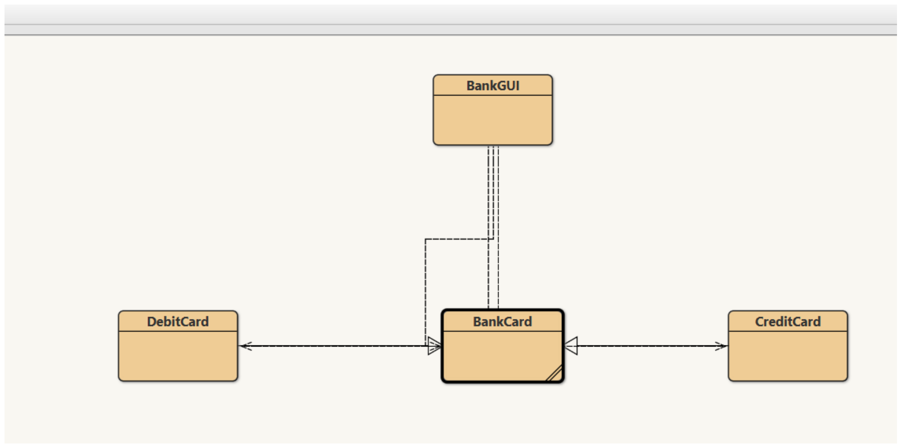


Figure 4 Classes interface in Blue J

3 Pseudo Code

Pseudo code is a simple and informal way of representing programming logic that is like a plain English. It's used to describe the basic steps of an algorithm or program without worrying about specific syntax or programming language rules.

Pseudo code is used as a tool for planning and designing programs, and it's often used as a communication tool between programmers and non-programmers. It can help to clarify the logic of a program and make it easier to understand, even for those without programming experience.

The structure of pseudo code is similar to that of a programming language, with statements and control structures such as loops and conditionals. However, pseudo code does not need to adhere to strict syntax rules, and variables and data types are often represented in plain English.

Pseudo code is a useful tool for planning and designing programs, as it allows programmers to focus on the logic of the program without worrying about syntax or language-specific details. It's a great way to communicate ideas and designs to others, and it can help to make programs more understandable and maintainable in the long run.

3.2 Pseudo Code of Bank_GUI

```
CREATE a JFrame object called frame1 with the title "BankGUI"
SET the size of frame1 to 450x500
SET the default close operation of frame1 to EXIT_ON_CLOSE
SET the layout of frame1 to null
SET the resizable property of frame1 to false
SET the location of frame1 to be centered

CREATE a JLabel object called cardId with the title "Card ID"
SET the bounds of cardId to 15* 15* 100* 30
ADD cardId to frame1

CREATE a JTextField object called textField1
SET the bounds of textField1 to 140* 15* 150* 30
ADD textField1 to frame1

CREATE a JLabel object called clientName with the title "Client Name"
SET the bounds of clientName to 15* 50* 100* 30
ADD clientName to frame1

CREATE a JTextField object called textField2
SET the bounds of textField2 to 140* 50* 150* 30
ADD textField2 to frame1

CREATE a JLabel object called issuerBank with the title "Issuer Bank"
SET the bounds of issuerBank to 15* 85* 100* 30
ADD issuerBank to frame1

CREATE a JTextField object called textField3
SET the bounds of textField3 to 140* 85* 150* 30
```

```
ADD textField3 to frame1

CREATE a JLabel object called bankAccount with the title "Bank Account"
SET the bounds of bankAccount to 15* 120* 100* 30
ADD bankAccount to frame1

CREATE a JTextField object called textField4
SET the bounds of textField4 to 140* 120* 150* 30
ADD textField4 to frame1

CREATE a JLabel object called balanceAmount with the title "Balance Amount"
SET the bounds of balanceAmount to 15* 155* 100* 30
ADD balanceAmount to frame1

CREATE a JTextField object called textField5
SET the bounds of textField5 to 140* 155* 150* 30
ADD textField5 to frame1

CREATE a JLabel object called pinNumber with the title "Pin Number"
SET the bounds of pinNumber to 15* 190* 100* 30
ADD pinNumber to frame1

CREATE a JTextField object called textField6
SET the bounds of textField6 to 140* 190* 150* 30
ADD textField6 to frame1

CREATE a JButton object called ADDdebit with the title "ADD Debit Card"
SET the bounds of ADDdebit to 15* 235* 140* 30
ADD ADDdebit to frame1
```

```
 ADD an action listener to ADDdebit
 WHEN ADDdebit is clicked
        CALL the ADDDebit method)
END
CREATE a JButton object called DisplayD with the title "Display"
SET the bounds of DisplayD to 165* 235* 100* 30
ADD DisplayD to frame1

Add ActionListener() to DisplayD
When Clicked
  CALL showDebit()
END
CREATE a JButton object called withdrawcard with the title "Withdraw Card"
SET the bounds of withdrawcard to 275* 235* 140* 30
ADD withdrawcard to frame1ADD clear to frame1

Add ActionListener() to withdrawcard
When Clicked
  Set frame2 visibility to true
  Dispose frame1
END
CREATE a JButton object called creditcard with the title "Credit Card"
SET the bounds of creditcard to 100* 270* 100* 30
ADD creditcard to frame1

Add ActionListener() to creditcard
When Clicked
  Set frame3 visibility to true
  Dispose frame1
END
CREATE a JButton object called clear with the title "Clear"
```

```
SET the bounds of clear to 225* 270* 100* 30
ADD clear to frame1

Add ActionListener() to clear
When Clicked
   Set textField1 text to ""
   Set textField2 text to ""
   Set textField3 text to ""
   Set textField4 text to ""
   Set textField5 text to ""
   Set textField6 text to ""
END

SET the visibility of frame1 to true

CREATE a new JFrame object* frame2
SET frame2 size to 450*500
SET frame2 default close operation to EXIT_ON_CLOSE
SET frame2 layout to null
SET frame2 resizable to false
SET frame2 location to center of the screen

CREATE a JLabel object named withdrawcardId with title "CardId"
SET the bounds of withdrawcardId to 15*15*100*30
ADD withdrawcardId to frame2

CREATE a JTextField object named textField7
SET the bounds of textField7 to 140*15*150*30
ADD textField7 to frame2

CREATE a JLabel object named withdrawpinNumber with title "Pin Number"
```

```
SET the bounds of withdrawpinNumber to 15*55*100*30
ADD withdrawpinNumber to frame2

CREATE a JTextField object named textField8
SET the bounds of textField8 to 140*55*150*30
ADD textField8 to frame2

CREATE a JLabel object named withdrawalAmount with title "Withdrawal Amount"
SET the bounds of withdrawalAmount to 15*95*100*30
ADD withdrawalAmount to frame2

CREATE a JTextField object named textField9
SET the bounds of textField9 to 140*95*150*30
ADD textField9 to frame2

CREATE a JLabel object named withdrawalDate with title "Withdrawal Date"
SET the bounds of withdrawalDate to 15*135*100*30
ADD withdrawalDate to frame2

CREATE a JComboBox<String> object named years with elements from the "year" array
SET the bounds of years to 140*135*60*30
ADD years to frame2

CREATE a JComboBox<String> object named months with elements from the "month" array
SET the bounds of months to 210*135*60*30
ADD months to frame2
```

```
**CREATE** a JComboBox<String> object named days with elements from the
"day" array
SET the bounds of days to 280*135*60*30
ADD days to frame2

CREATE a JButton object named withdraw with title "Withdraw"
SET the bounds of withdraw to 50*175*100*30
ADD withdraw to frame2

Add ActionListener() to withdraw
When Clicked
   Call Withdraw()
END

CREATE a JButton object named withdarwclear with title "clear"
SET the bounds of withdarwclear to 160*175*100*30
ADD withdarwclear to frame2

Add ActionListener() to withdrawclear
When Clicked
   SET textField7 text to ""
   SET textField8 text to ""
   SET textField9 text to ""
   SET years selected index to 0
   SET months selected index to 0
   SET days selected index to 0
END
```

```
CREATE a JButton object named withdrawback with title "Back"
SET the bounds of withdrawback to 270*175*100*30
ADD withdrawback to frame2
Add ActionListener() to withdrawback
When Clicked
    SET frame1 visibility to true
    Dispose frame2
END

CREATE a JFrame object named frame3
SET the size of frame3 to 500*600
SET the default close operation of frame3 to JFrame.EXIT_ON_CLOSE
SET the layout of frame3 to null
SET the resizable property of frame3 to false
SET the location of frame3 to the center of the screen

SET the visibility of frame3 to true

CREATE JLabel object named creditcardId with title "CardId"
SET the bounds of creditcardId to 15*15*100*30
ADD creditcardId to frame3

CREATE JTextField object named textField12
SET the bounds of textField12 to 140*15*150*30
ADD textField12 to frame3

CREATE JLabel object named creditClientName with title "Client Name"
SET the bounds of creditClientName to 15*55*100*30
ADD creditClientName to frame3

CREATE JTextField object named textField15
```

```
**SET** the bounds of textField15 to 140*55*150*30
**ADD** textField15 to frame3

**CREATE** JLabel object* InterestRate with title “Interest Rate”
**SET** InterestRate bounds 15*95*100*30
**ADD** InterestRate in frame3

**CREATE** JTextField object* textField13
**SET** textField13 bounds 140*95*150*30
**ADD** textField13 in frame3

**CREATE** JLabel object* creditIssuerBank with title "Issuer Bank"
**SET** creditIssuerBank bounds 15*135*100*30
**ADD** creditIssuerBank to frame3

**CREATE** JTextField object* textField16
**SET** textField16 bounds 140*135*150*30
**ADD** textField16 to frame3

**CREATE** JLabel object* creditBankAccount with title "Bank Account"
**SET** creditBankAccount bounds 15*175*100*30
**ADD** creditBankAccount to frame3

**CREATE** JTextField object* textField17
**SET** textField17 bounds 140*175*150*30
**ADD** textField17 to frame3

**CREATE** JLabel object* creditBalanceAmount with title “Balance Amount”
**SET** creditBalanceAmount bounds 15*215*100*30
**ADD** creditBalanceAmount in frame3
```

```
CREATE JTextField object* textField18
SET textField18 bounds 140*215*150*30
ADD textField18 in frame3

CREATE JLabel object* expirationDate with title “Expiration Date”
SET expirationDate bounds 15*255*100*30
ADD expirationDate in frame3

CREATE JComboBox object* years2 with options from year array
SET years2 bounds 140*255*90*30
ADD years2 in frame3

CREATE JComboBox object* months2 with options from month array
SET months2 bounds 240*255*90*30
ADD months2 in frame3

CREATE JComboBox object* days2 with options from day array
SET days2 bounds 340*255*90*30
ADD days2 in frame3

CREATE JLabel object* CVCNumber with title "CVC Number"
SET CVCNumber bounds to 15*295*100*30
ADD CVCNumber in frame3
CREATE JTextField object* textField10
SET textField10 bounds to 140*295*150*30
ADD textField10 in frame3

CREATE JLabel object* creditLimit with title "Credit Limit"
SET creditLimit bounds 15*335*100*30
ADD creditLimit in frame3
```

```
**CREATE** JTextField object* textField11
**SET** textField11 bounds 140*335*150*30
**ADD** textField11 in frame3

**CREATE** JLabel object * graceperiod with title “Grace Period”
**SET** graceperiod bounds 15*375*100*30
**ADD** graceperiod in frame3

**CREATE** JTextField object* textField14
**SET** textField14 bounds 140*375*150*30
**ADD** textFeild14 in frame3

**CREATE** JButton object* addcredit with title “addcaredit card”
**SET** addcredit bounds 30*415*150*30
**ADD** addcredit in frame3

**Add ActionListener() to addcredit**
When Clicked
   **CALL** addCreditcard()
**END**

**CREATE** JButton object* back with text “Back”
**SET** back bounds 190*415*100*30
**ADD** back in frame3

**Add** ActionListener() to creditclear
When Clicked
   **Set** textField10 text to ""
   **Set** textField11 text to ""
   **Set** textField12 text to ""
   **Set** textFeild13 text to ""
```

```
    **Set** textField14 text to ""
    **Set** years2 selected index to 0
    **Set** months2 selected index to 0
    **Set** days2 selected index to 0
END
CREATE JButton object* creditclear with title “clear”
SET creditclear bounds 300*415*100*30
ADD creditclear in frame3

CREATE JButton object* Displayc with title “Display”
SET Displayc bounds 10*455*150*30
ADD Displayc in frame3

Add ActionListener() to Displayc
When Clicked
    CALL displayCredit()
END

CREATE JButton object* ADDcreditLimit with text “ADD Credit Limit”
SET addcreditLimit bounds 170*455*150*30
ADD addcreditLimit in frame3

Add ActionListener() to addcreditLimit
When Clicked
    Call creditLimit()
END

CREATE JButton object* cancelcredit with title “Cancel credit Card”
SET cancelcredit bounds 330*455*150*30
ADD cancelcredit in frame3
Add ActionListener() to cancelcredit
```

```
        When Clicked
          Call cancelCreditCard()
        END

FUNCTION getCardID()

    CREATE int variable, CardId and INITIALIZE CardId to INVALID

    TRY

        CREATE integer variable, CardId and INITILIZE CardId to Integer.parseInt(textField1.getText().trim())

        IF CardId is less than or equal to 0

            SET CardId to INVALID

        END IF

    CATCH(Exception ae)

        Show message dialog box with error message

    END TRY

    RETURN CardId

END FUNCTION


FUNCTION getClientName()

    CREATE String variable, Name and INITILIZE Name to clientName.getText()

    RETURN Name
```

```
END FUNCTION

FUNCTION getIssuerBank()
    RETURN textField2.getText().trim()
END FUNCTION

FUNCTION getBankAccount()
    RETURN textField4.getText().trim()
END FUNCTION

FUNCTION getBalanceAmount()
    CREATE int variable, balanceAmount and INITILIZE balanceAmount to INVALID
    TRY
        CREATE integer variable, balanceAmount and INITILIZE balanceAmount
        to Integer.parseInt(textField5.getText().trim())
        IF balanceAmount is less than or equal to 0
            SET balanceAmount to INVALID
        END IF
    CATCH(Exception ae)
```

```
        Show message dialog box with error message
    END TRY
    RETURN balanceAmount
END FUNCTION


FUNCTION getPinNumber()
    CREATE int variable, pinNumber and INITILIZE pinNumber to INVALID
    TRY
        CREATE integer variable, pinNumber and INITILIZE pinNumber to
Integer.parseInt(textField6.getText().trim())
        IF pinNumber is less than or equal to 0
            SET pinNumber to INVALID
        END IF
    CATCH(Exception ae)
        Show message dialog box with error message
    END TRY
    RETURN pinNumber
END FUNCTION
FUNCTION checkCardIdUnique(cardId)
```

```
        CREATE boolean variable, result and INITIALIZE result to true

        FOR each BankCard object in array DO

                IF the object is an instance of DebitCard THEN

                        CREATE a DebitCard variable, list, and CAST the object to
                        DebitCard

                        IF the list's cardId is equal to cardId THEN

                                SET result to false

                        END IF

                END IF

        END FOR

        RETURN result

END FUNCTION


FUNCTION checkCreditCardIdUnique(cardId)

   CREATE boolean variable, result and INITIALIZE result to true

   FOR each obj IN array

      IF obj is instance of CreditCard THEN

         CREATE CreditCard variable, list and SET list to obj

         IF list.getCardId() is equal to cardId THEN
```

```
            SET result to false
         END IF
      END IF
   END FOR
   RETURN result
END FUNCTION
FUNCTION showDebit()
     FOR each obj in array
          IF obj is instance of DebitCard
               CREATE DebitCard variable, debitCard, and ASSIGN obj to debitCard
               CALL debitCard.display()
          END IF
     END FOR
END FUNCTION


FUNCTION addDebit()
     CREATE string variable, clientName and ASSIGN value RETURNed by getClientName()
```

```
    CREATE string variable, issuerBank and ASSIGN value RETURNed by getIssuerBank()

    CREATE string variable, bankAccount and ASSIGN value RETURNed by getBankAccount()

    CREATE integer variable, cardId and ASSIGN value RETURNed by getCardID()

        CREATE double variable, balanceAmount and ASSIGN value returned by getBalanceAmount()

    CREATE integer variable, pinNumber and ASSIGN value Returned by getPinNumber()


    IF clientName is empty OR issuerBank is empty OR bankAccount is empty

        IF clientName is empty

            Show message dialog box with error message to enter client name

            RETURN

        END IF


        IF issuerBank is empty

            Show message dialog box with error message to enter issuer bank name

            RETURN

        END IF
        IF bankAccount is empty
            Show message dialog box with error message to enter bank account
name
            RETURN
        END IF
    END IF


    IF checkDebitCardUniqueCardId(cardId) is true
        ADD new DebitCard object to list with given parameters
        Show message dialog box with success message
    ELSE
        Show message dialog box with error message to enter unique card Id
    END IF
END FUNCTION
FUNCTION getWCardId()
  CREATE int variable, cardId and INITIALIZE cardId to INVALID
  TRY
    CREATE integer variable, cardId and INITILIZE cardId to textField7.getText().trim()
```

```
        IF cardId is less than or equal to 0
            SET cardId to INVALID
        END IF
    END TRY
    CATCH(Exception ae)
        Show message dialog box with error message
    END CATCH
    RETURN cardId
END FUNCTION
FUNCTION getWithdrawAmount()
        CREATE integer variable, withdrawamount and INITIALIZE withdrawamount to INVALID
        TRY
                CREATE integer variable, withdrawamount and INITILIZE withdrawamount to textField8.getText().trim()
                IF withdrawamount is less than or equal to 0
                        SET withdrawamount to INVALID
                END IF
        END TRY
        CATCH(Exception ae)
```

```
                Show message dialog box with error message

        END CATCH

        RETURN withdrawamount

END FUNCTION



FUNCTION getwPinnumber()

   CREATE int variable, pinNumber and INITIALIZE pinNumber to INVALID

   TRY

      CREATE integer variable, pinNumber and INITILIZE pinNumber to textField9.getText().trim()

      IF pinNumber is less than or equal to 0

         SET pinNumber to INVALID

      END IF

   END TRY

   CATCH(Exception ae)

      Show message dialog box with error message

   END CATCH

   RETURN pinNumber

END FUNCTIO
```

```
FUNCTION getWithdrawalDate()

    CREATE empty string variable, date

    SET year to years.getSelectedItem().toString()

    SET month to months.getSelectedItem().toString()

    SET day to days.getSelectedItem().toString()


    IF year equals "year" OR month equals "month" OR day equals "day" THEN

        SET date to null

    ELSE

        SET date to (year + "-" + month + "-" + day)

    END IF


    RETURN date

END FUNCTION


FUNCTION Withdraw()

    CREATE integer variable, pinNumber and ASSIGN value RETURNed by
    getwPinnumber()

    CREATE integer variable, withdrawAmount and ASSIGN value RETURNed by
    getWithdrawAmount()
    CREATE string variable, withdrawalDate and ASSIGN value RETURNed by
    getWithdrawalDate()

    CREATE integer variable, cardId and ASSIGN value RETURNed by getWCardId()


    IF getWithdrawAmount() is equal to 0

        Show message dialog box with error message

        RETURN

    END IF


    IF getwPinnumber() is less than or equal to 0 OR getPinNumber() is not equal to
getwPinnumber()

        Show message dialog box with error message

        RETURN

END IF


    IF getWithdrawalDate() is null

        Show message dialog box with error message

        RETURN

    END IF
    IF getWCardId() is equal to 0
        Show message dialog box with error message
        RETURN
    END IF


    FOR each obj in array
        IF obj is an instance of DebitCard
            CREATE DebitCard variable, debitCard and ASSIGN obj
            IF debitCard's cardId is equal to cardId
                CALL debitCard's withdraw FUNCTION with withdrawAmount, pinNumber, and withdrawalDate as arguments
                Show message dialog box with success message
            ELSE
                Show message dialog box with warning message
            END IF
        END IF
    END FOR
END FUNCTION
```

```
FUNCTION showCredit()
        FOR EACH obj in array
                IF obj is an instance of CreditCard THEN
                        CREATE CreditCard variable creditCard and CAST obj to CreditCard
                        CALL creditCard.display()
                END IF
        END FOR
END FUNCTION
FUNCTION getCreditCardID()
   CREATE int variable, cardid and INITIALIZE cardid to INVALID
   TRY
        CREATE integer variable, cardid and INITIALIZE cardid to
        textField12.getText().trim()
      IF cardid is less than or equal to 0
         SET cardid to INVALID
      END IF
   END TRY
   CATCH(Exception ae)
      Show message dialog box with error message
```

```
    END CATCH

    RETURN cardid

END FUNCTION



FUNCTION getCreditClientName()

    RETURN trimmed textField15 text

END FUNCTION



FUNCTION getCreditIssuerBank()

    RETURN trimmed textField16 text

END FUNCTION



FUNCTION getCreditBankAccount()

    RETURN trimmed textField17 text

END FUNCTION

FUNCTION getCreditBalanceAmount()

    CREATE integer variable, balanceAmount and INITIALIZE balanceAmount to 0

    TRY
```

```
        CREATE integer variable, balanceAmount and INITILIZE balanceAmount to textField18.getText().trim()
        IF balanceAmount is less than 0
            SET balanceAmount to 0
        END IF
    END TRY
    CATCH(Exception ae)
        Show message dialog box with error message
    END CATCH
    RETURN balanceAmount
END FUNCTION


FUNCTION getCVCNumber()
    CREATE integer variable CVCNumber and INITIALIZE it to INVALID
    TRY
        CREATE integer variable CVCNumber and INITIALIZE it to textField10.getText()
        IF CVCNumber is less than 0
            SET CVCNumber to INVALID
        END IF
    END TRY
    CATCH(Exception ae)
        Show message dialog box with error message
    END CATCH
    RETURN CVCNumber
END FUNCTION
FUNCTION getInterestRate()
        CREATE double variable, interestRate and INITIALIZE interestRate to INVALID
        TRY
                CREATE double variable, interestRate and INITIALIZE interestRate to Double.parseDouble(textField13.getText())
                IF interestRate is less than or equal to 0
                        SET interestRate to INVALID
                END IF
        END TRY
        CATCH (Exception ae)
                Show message dialog box with error message
        END CATCH
        RETURN interestRate
```

```
END FUNCTION

FUNCTION getExpirationDate()
    CREATE string variables year, month, day, date
    SET year to the selected item from years2 combobox
    SET month to the selected item from months2 combobox
    SET day to the selected item from days2 combobox

    IF year equals "year" OR month equals "month" OR day equals "day" THEN
        SET date to null
    ELSE
        SET date to concatenation of year, "-", month, "-", and day
    END IF

    RETURN date
END FUNCTION
FUNCTION addCredit()
        CREATE integer variable, creditCardId and ASSIGN value RETURNed by
        getCreditCardId()
```

```
CREATE string variable, creditClientName and ASSIGN value RETURNed by getCreditClientName()

CREATE integer variable, creditBalanceAmount and ASSIGN value RETURNed by getCreditBalanceAmount()

CREATE string variable, creditBankAccount and ASSIGN value RETURNed by getCreditBankAccount()

CREATE double variable, interestRate and ASSIGN value RETURNed by getInterestRate()

CREATE string variable, creditIssuerBank and ASSIGN value RETURNed by getCreditIssuerBank()

CREATE integer variable, CVCNumber and ASSIGN value RETURNed by getCVCNumber()

CREATE string variable, expirationDate and ASSIGN value RETURNed by getExpirationDate()


IF creditCardId is less than or equal to 0

        Show message dialog box with error message

        RETURN

END IF


IF creditClientName is empty

        Show message dialog box with error message
        RETURN
    END IF

    IF creditBalanceAmount is less than 0
        Show message dialog box with error message
        RETURN
    END IF

    IF creditIssuerBank is empty
        Show message dialog box with error message
        RETURN
    END IF

    IF checkCreditCardUniqueCardId(creditCardId) is true
        Add new CreditCard object to list with given parameters
        Show message dialog box with success message
    END IF
    ELSE
        Show message dialog box with error message
    END ELSE
END FUNCTION
FUNCTION displayCredit()
    FOR EACH obj in array DO
        IF obj is instance of CreditCard THEN
            CREATE CreditCard variable creditCard and CAST obj to CreditCard
            CALL creditCard.display()
        END IF
    END FOR
END FUNCTION


FUNCTION getCardid()
    CREATE int variable, cardid and INITIALIZE cardid to INVALID
    TRY
        CREATE integer variable, cardid and INITILIZE cardid to textField12.getText().trim()
        IF cardid is less than or equal to 0
            Show message dialog box with inFORmation message "CardId cannot be less than 1"
        END IF
    END TRY
    CATCH(Exception ae)
        Show message dialog box with error message "INVALID INPUT \n Please enter valid Card ID"
    END CATCH
    RETURN cardid
END FUNCTION
FUNCTION getCreditLimit()
    CREATE double variable, creditLimit and INITIALIZE creditLimit to INVALID
    TRY
        CREATE double variable, creditLimit and INITIALIZE creditLimit to Double.parseDouble(textField18.getText().trim())
        IF creditLimit is less than or equal to 0
            SET creditLimit to INVALID
        END IF
    CATCH(Exception ae)
        Show message dialog box with error message
    END TRY
    RETURN creditLimit
END FUNCTION
```

```
FUNCTION getGracePeriod()
    CREATE integer variable, graceperiod and INITIALIZE graceperiod to INVALID
    TRY
        CREATE integer variable, graceperiod and INITIALIZE graceperiod to Integer.parseInt(textField14.getText())
        IF graceperiod is less than or equal to 0
            SET graceperiod to INVALID
        END IF
    END TRY
    CATCH (Exception ae)
        Show message dialog box with error message
    END CATCH
    RETURN graceperiod
END FUNCTION
FUNCTION addCreditLimit()
    CREATE integer variable, limitCardId and ASSIGN value RETURNed by getLimitCardId()
    CREATE integer variable, creditCardId and ASSIGN value RETURNed by getCreditCardId()
    CREATE double variable, creditlimit and ASSIGN value RETURNed by getCreditLimit()

    CREATE integer variable, gracePeriod and ASSIGN value RETURNed by getGracePeriod()


    IF limitCardId is equal to INVALID OR limitCardId is not equal to creditCardId THEN
        Show message dialog box with inFORmation message
        RETURN
    END IF


    IF creditlimit is equal to INVALID THEN
        Show message dialog box with inFORmation message
        RETURN
    END IF


    IF gracePeriod is equal to INVALID THEN
        Show message dialog box with inFORmation message
        RETURN
    END IF
```

```
        FOR each BankCard obj in list
            IF obj is instance of CreditCard THEN
                    CREATE CreditCard variable, creditCard and ASSIGN obj
                     IF creditCard's card ID is equal to limitCardId THEN
                    SET creditCard's credit limit to creditlimit AND grace period to
                    gracePeriod
                    Show message dialog box with inFORmation message
                ENF IF
                     ELSE
                     Show message dialog box with warning message
                 END ELSE
            END IF
          END FOR
END FUNCTION
FUNCTION getCancelCardId()
        CREATE int variable, cardId and INITIALIZE cardId to INVALID
        TRY
            CREATE integer variable, cardId and INITILIZE cardId to
textField12.getText()
```

```
        IF cardId is less than or equal to 0
            SET cardId to INVALID
        END IF
    END TRY
    CATCH(Exception e)
        Show message dialog box with error message
    END CATCH
    RETURN cardId
END FUNCTION
FUNCTION cancelCreditCard()
    SET boolean variable isfound to false
    SET integer variable cardID to the result of calling getCancelCardId()
    FOR each BankCard object in array DO
        IF the object is an instance of CreditCard THEN
            CREATE CreditCard variable creditCard and ASSIGN the object to it
            IF the creditCard's cardId is equal to cardID THEN
                SET isfound to true
                CALL creditCard's cancelCreditCard method
                BREAK the loop
            **END IF**

        **END IF**

    **END FOR**

    **IF** isfound is true THEN

        Show message dialog box with "Canceled credit card" message and "success box" title

    **ELSE**

        Show message dialog box with "cardId doesnt match" message and "error Box" title

    **END IF**

**END FUNCTION**
```

4 Method Description

Method Name	Descreiption
getCardId() :	This accessor function returns the value of the BankCard's CardId instance variable Using the getText (), the cardId&#39;s information are obtained.
getClientName():	This accessor method returns the iThis accessor function and returns the instance variable Client name for the Bank card as its value. Using getText (), the client name's information is captured.
getBalanceAmount()	This accessor method returns the instance variable with the value Bank card Balance Amount. Using getText (), the balance amount's specifics are obtained
getPinNumber()	This accessor method retrieves the value of the debit card's instance variable pin number, which is used to withdraw the remaining money.
add Debit()	This function adds a debit card to an array list of Bank card class and has void as the return type.
Show Debit()	This access modifier display the data of debitt card which is stored in array list.
getWCardId()	This function takes the cardId from the withdrawal frame and checks if it is integer form or not.

getWithdrawAmount()	The value instance variable withdrawal amount of the Debit card, which is used to withdraw balance amount, is returned by this accessor method. Using getText (), the withdrawal amount's specifics are captured.
getwPinnumber()	This accessor method retrieves the debit card's value instance variable pin number, which is needed to withdraw the remaining money. Using getText (), the pin number's data are captured.
getWithdrawalDate()	This accessor method provides the date of withdrawal from the debit card that was used to withdraw the balance amount. Using getText (), the withdrawal date's specifics are captured.
Withdrawal()	This technique allows you to withdraw money from your debit card and update your balance while using void as the return type.Amount and withdrawalWhen the submitted values are accepted, amount following withdrawal.

showCredit()	This function adds a credit card to an array list of bank cards of the class by returning void as the return type.
getCreditCardID() :	This accessor function returns the value of the BankCard's CardId instance variable Using getText (), the cardId's information are obtained.

getCreditClientName()	This accessor method returns the iThis accessor function and returns the instance variable Client name for the Bank card as its value. Using getText (), the client name's information is captured.
getCreditIssuerBank()	This accessor method returns the iThis accessor method returns the value of the BankCard instance variable Issuer bank. Using getText (), the Issuer bank's information is obtained.
getCreditBankAccount()	This accessor method returns the iThis accessor method returns the instance variable value for the Bank Account of the Bank card. Using getText (), the bank account's data are obtained.
getCreditBalanceAmount()	This accessor method returns the instance variable with the value Bank card Balance Amount. Using getText (), the balance amount's specifics are obtained.
getCVCNumber()	This accessor method delivers the credit card's CVC number, which is used to add credit cards, as the value for the instance variable. Using getText (), the CVCnumber's information is obtained.
getInterestRate()	This accessor method retrieves the credit card's value instance variable interest rate, which is utilized when adding credit cards. Using getText (), the interest rate's specifics are obtained.

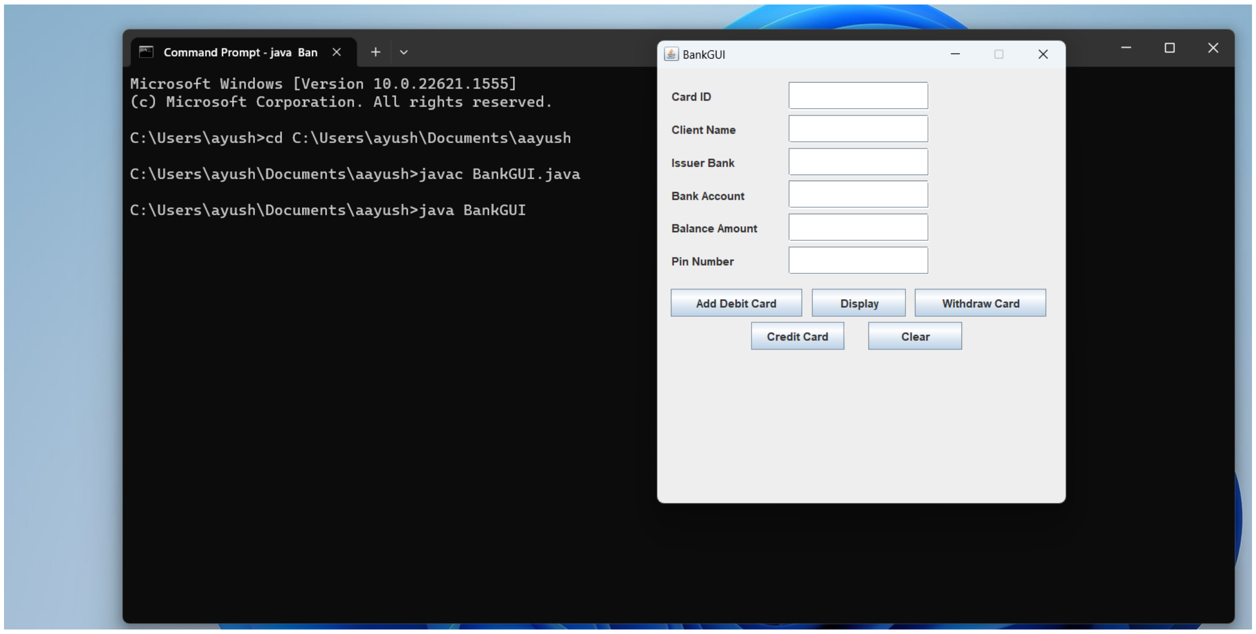
getExpirationDate()	The value of the cbinstance variable expiration date of the credit card that is used to add credit card is returned by this accessor method. Using getText (), the information of the expiration date are captured.
addCreditcard()	This function adds a credit card values to an array list of Bank card class and has void as the return type.
displayCredit()	This access modifier displays the data of credit card which is stored in array list.
getCardId()	This accessor function returns the value of the BankCard's CardId instance variable Using getText (), the cardId's information are obtained.
getCreditLimit()	This accessor method returns the instance variable CreditLimit's value, which is used to establish the credit card limit. Utilizing getText (), the credit limit's specifics are obtained.
getGracePeriod()	This accessor function returns the instance variable grace period's value, which is used to establish the credit card's credit limit. Using getText (), the grace period's specifics are captured.
creditLimit()	This accessor method returns the instance variable CreditLimit's value, which is used to establish the credit card limit. Utilizing getText (), the credit limit's specifics are obtained.
getCancelCardId()	This method, which cancels the credit card, returns void as its return type. This calls the CreditCard class's method for canceling the credit card.
cancelCreditCard()	This method, which cancels the credit card, returns void as its return type. This calls the CreditCard class's method for canceling the credit card.

5 Testing

5.1 Test 1: Compiling and running Java code using Terminal.

Table 1 table of test 1

Objective	To compile and run program using command prompt / terminal.
Action	Open command prompt and type command javac <name of .java file>. Javac is used to compile program and now type java <name of .java file>* this command is used after compiling which runs the main method of the program.
Expected result	Program would be simply compiled and run.
Actual Result	Program is compiled and run without error
Conclusion	Program can be compiled and run using command prompt.



```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ayush>cd C:\Users\ayush\Documents\aayush

C:\Users\ayush\Documents\aayush>javac BankGUI.java

C:\Users\ayush\Documents\aayush>java BankGUI
```

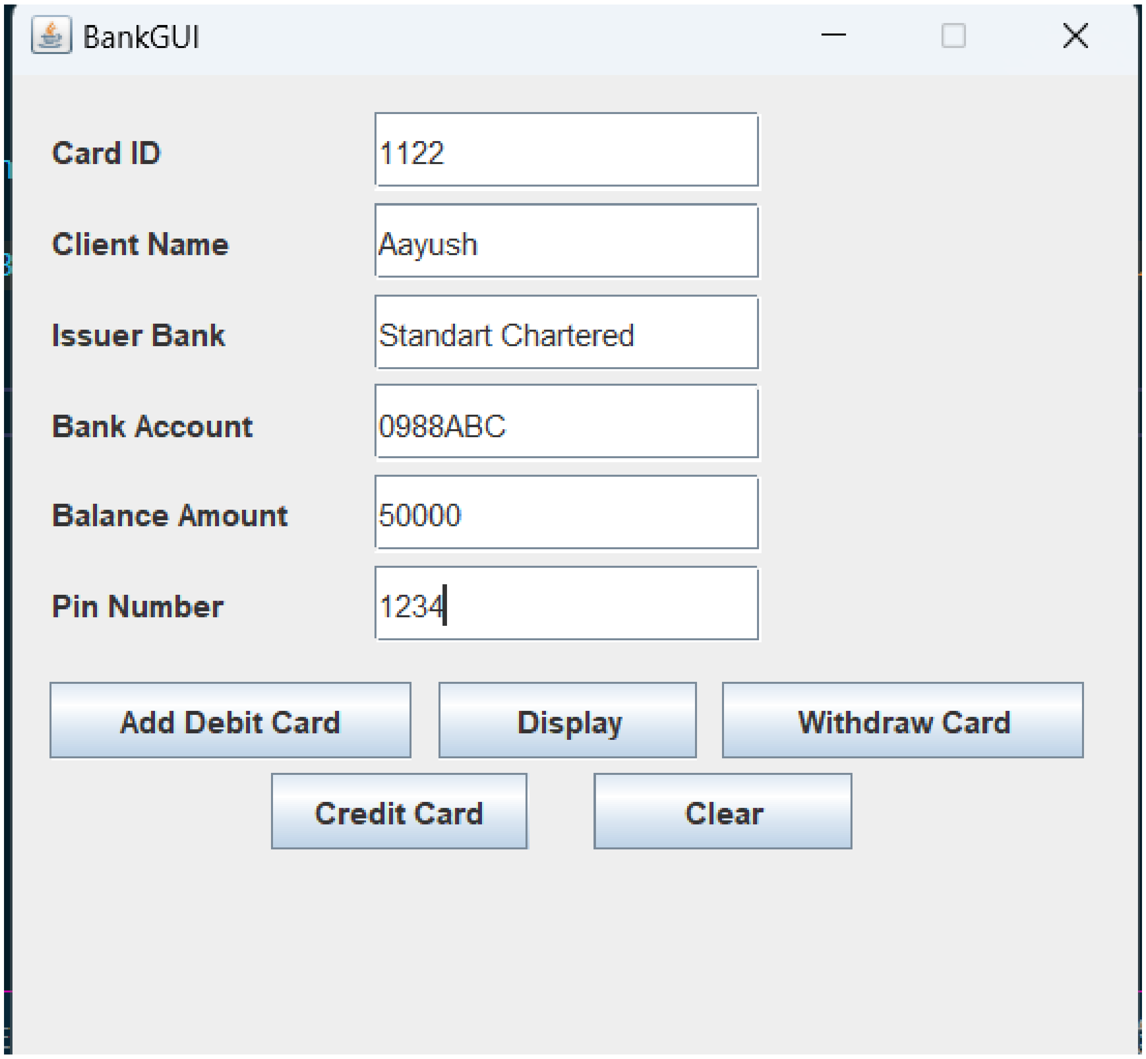
Figure 5 Evidence of Test 1

5.2 Test 2

(a) ADD Debit Card

Table 2 table of testing of adding debit card

Objective	**ADD** values to debit card class
Action	1. Open Bank GUI class and run it 2. Fill the details in the Debit Card frame. Card ID = 1122 Client Name = Aayush Issuer Bank = Standard Chartered Bank Account = 0988ABC Balance Amount = 50000 Pin Number = 1234 3. Click the **ADD** Debit Card Button
Expected Result	After clicking **ADD** Debit Card Button the message should be shown saying “DEBIT CARD **ADD**ED!”
Actual Result	The Debit Card is **ADD**ed as expected successfully.
Conclusion	The test is successful.



Card ID	1122
Client Name	Aayush
Issuer Bank	Standart Chartered
Bank Account	0988ABC
Balance Amount	50000
Pin Number	1234

Figure 6 Adding Value in Debit Card Class

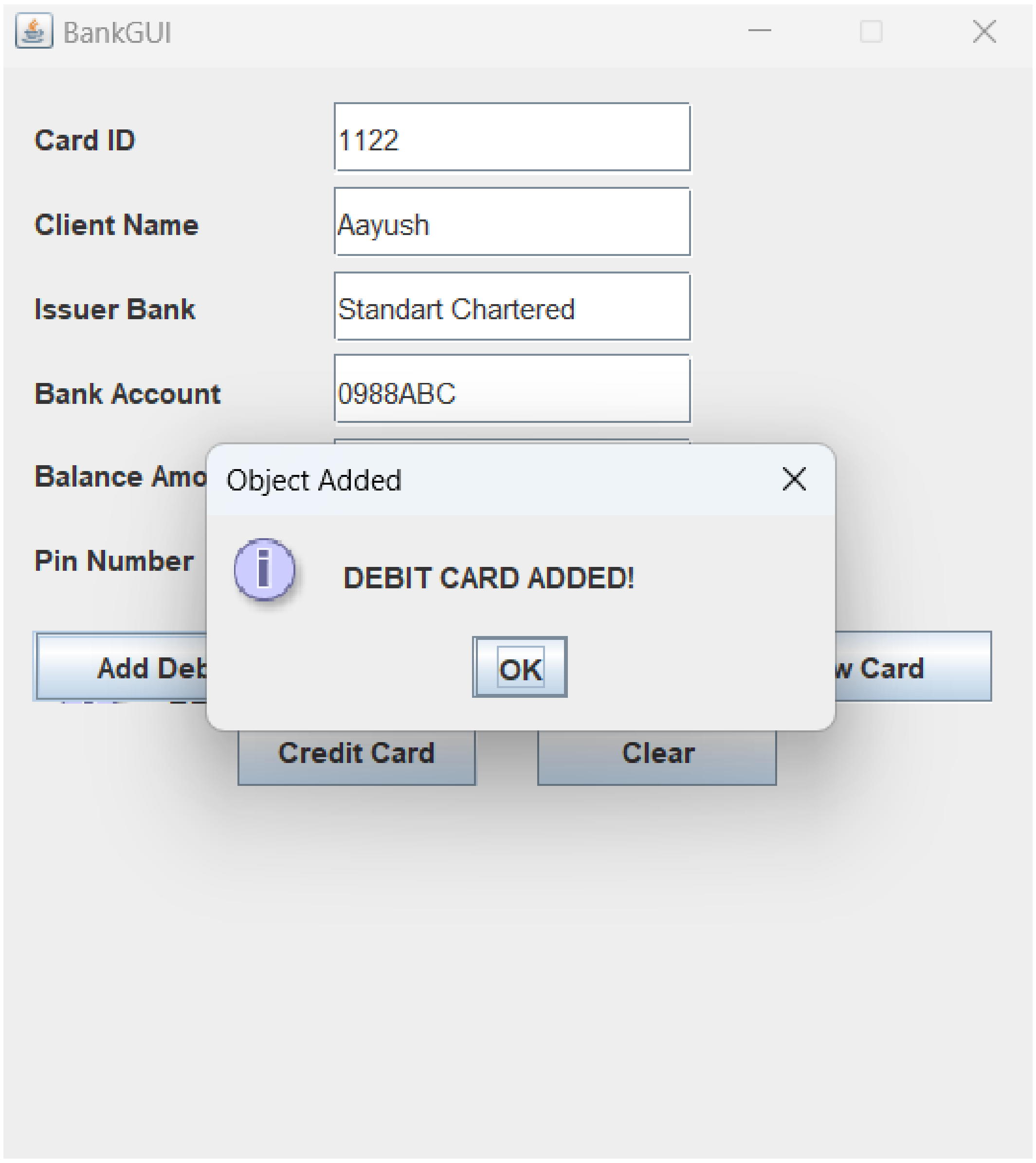
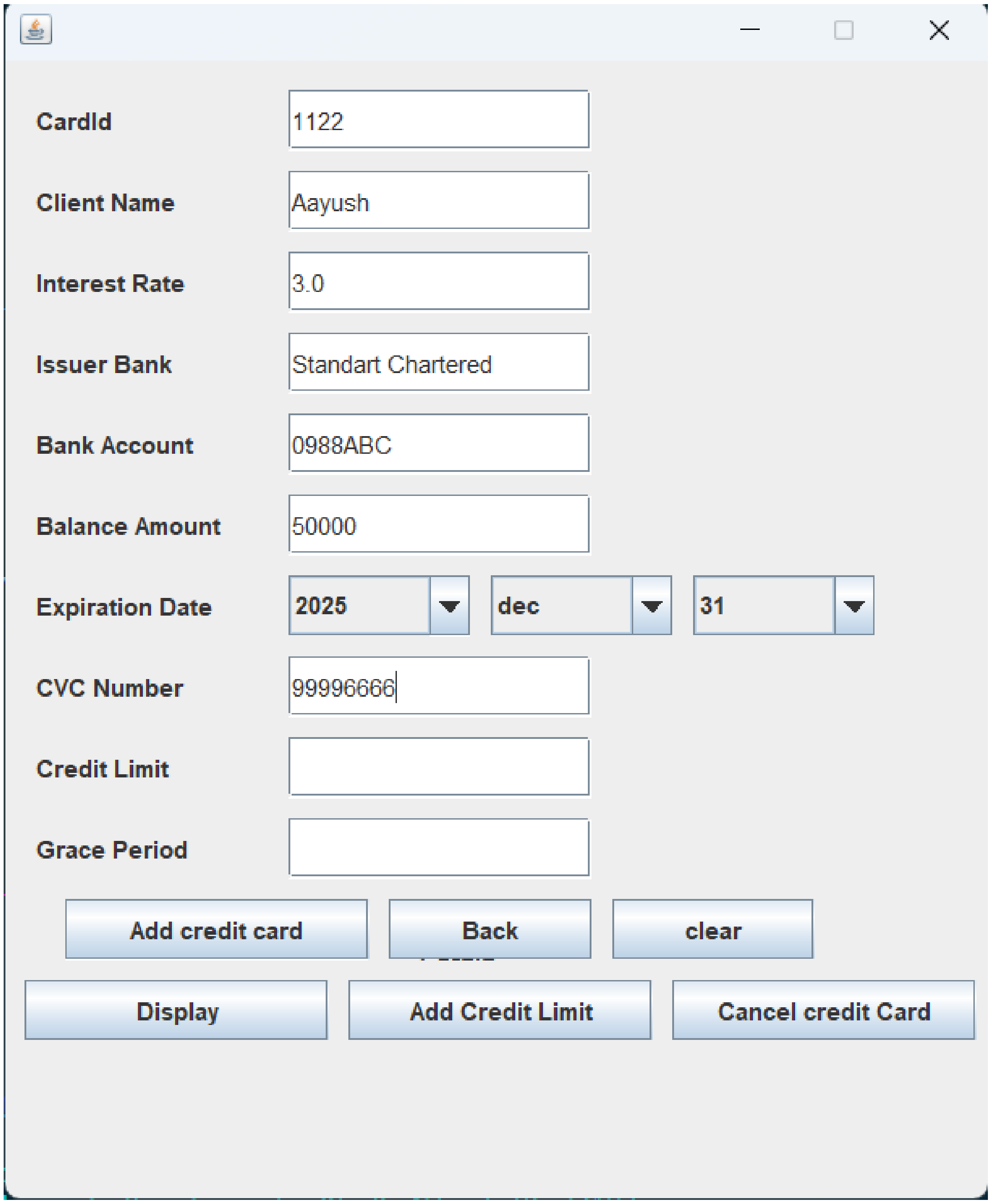


Figure 7 Popup message after adding values in Debit Card

(b) ADD Credit Card

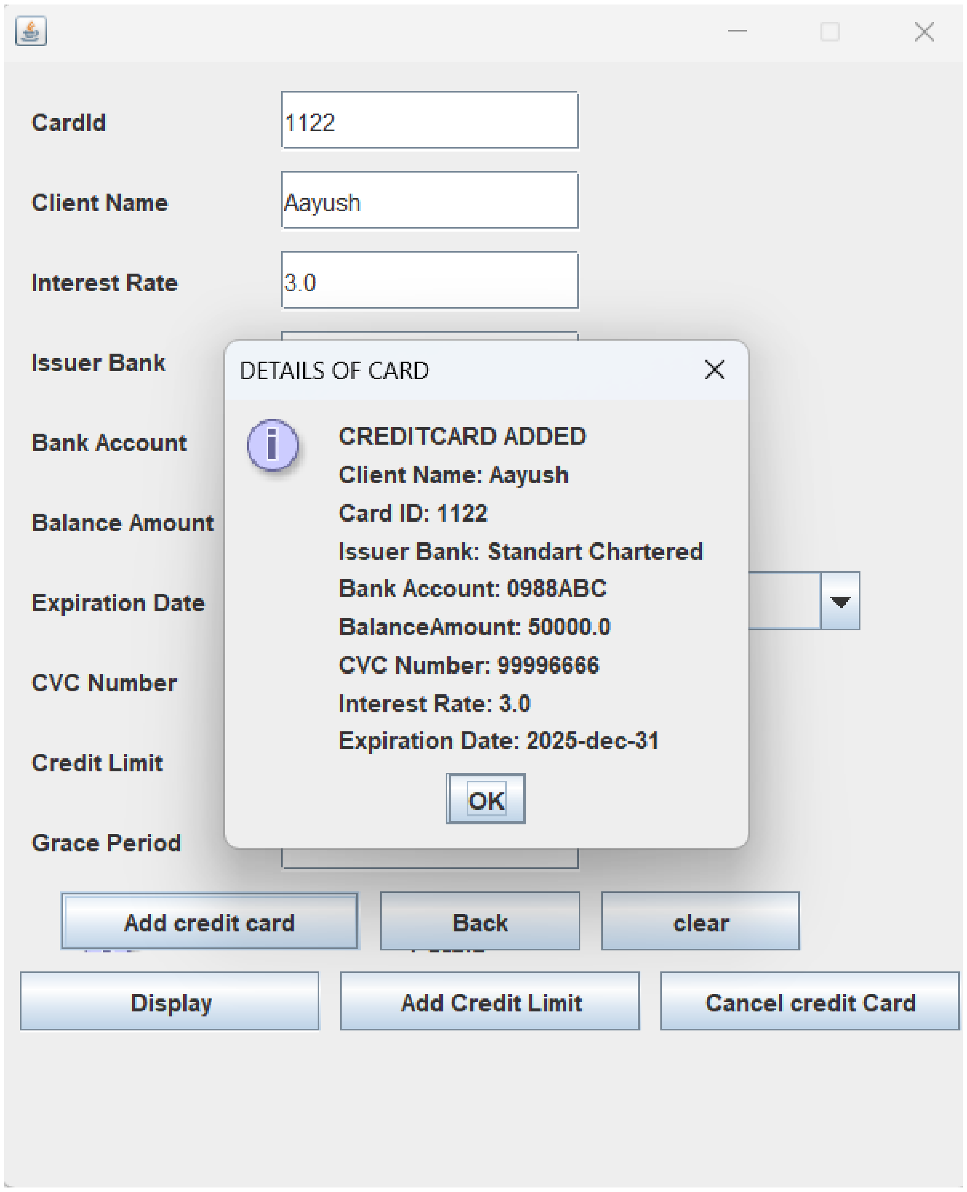
Table 3 table of testing of adding credit card

Objective	**ADD**ing values to Credit Card class
Action	Opening Credit Card Class and inputting the following values: 1. Card ID = 1122 2. Client Name = Aayush 3. Interest Rate = 3.0 4. Issuer Bank = Standard Chartered 5. Bank Account = 0988ABC 6 Balance Amount = 50000 7 Expiration Date = 2025 dec 31 8 CVC Number = 99996666
Expected Result	After clicking **ADD** Credit Card Button the message should be shown saying “Credit CARD **ADD**ED with displaying the values of: Client Name = Aayush Card ID = 1122 Issuer Bank = Standard Chartered Bank Account = 0988ABC Balance Amount = 50000 CVC Number = 99996666 Interest Rate = 3.0 Expiration Date = 2025 dec 31”
Actual Result	The values of Credit Card **ADD**ed and the popup message was shown as expected.
Conclusion	The test was successful.



CardId	1122
Client Name	Aayush
Interest Rate	3.0
Issuer Bank	Standart Chartered
Bank Account	0988ABC
Balance Amount	50000
Expiration Date	2025 dec 31
CVC Number	99996666
Credit Limit	
Grace Period	

Figure 8 Adding value in Credit Card class.



DETAILS OF CARD

CREDITCARD ADDED
Client Name: Aayush
Card ID: 1122
Issuer Bank: Standart Chartered
Bank Account: 0988ABC
BalanceAmount: 50000.0
CVC Number: 99996666
Interest Rate: 3.0
Expiration Date: 2025-dec-31

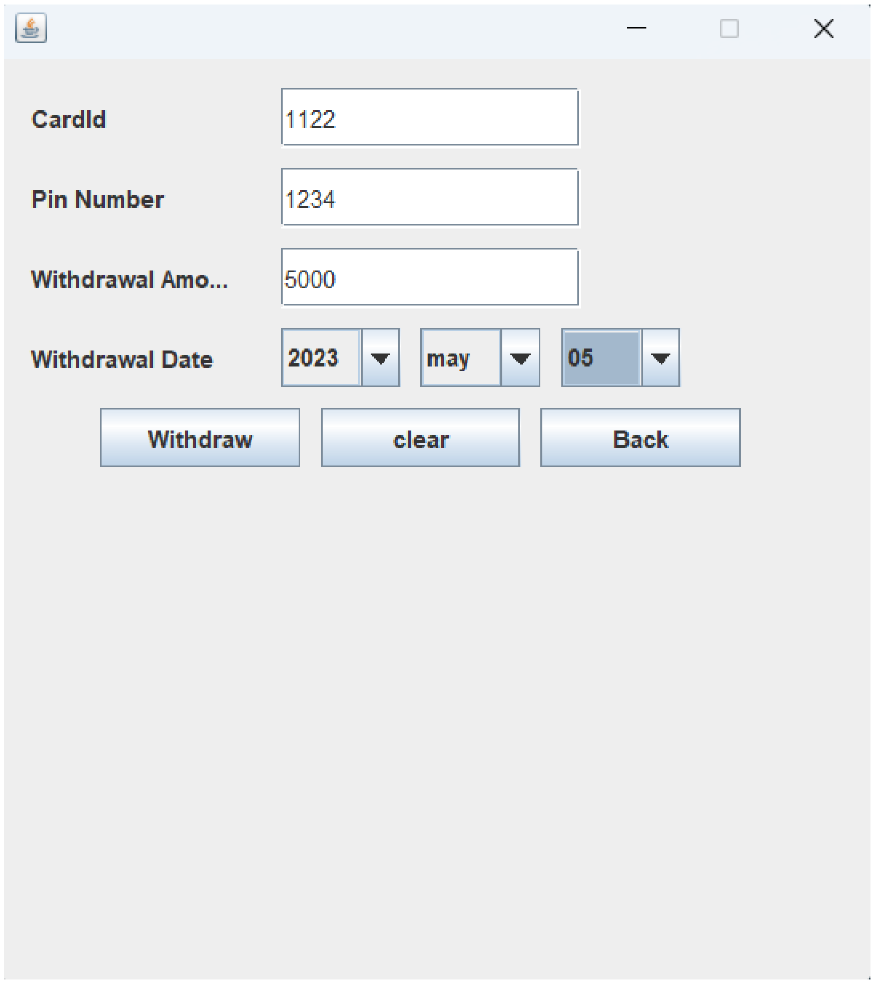
OK

Figure 9 Popup message shown after adding values in Credit Card class.

(c) Withdraw amount from Debit Card

Table 4 table of withdraw amount from debit card

Objective	Successfully Withdraw Amount from Debit Card Class
Action	Inputting the values of: Card ID = 1122 Pin Number = 1234 Withdrawal Amount = 5000 Withdrawal Date = 2023* may* 05
Expected Result	When clicking withdraw a message should popup saying “Withdrawal Successful”
Actual Result	The withdrawal was successful when we clicked withdrawal button.
Conclusion	The test was successful.



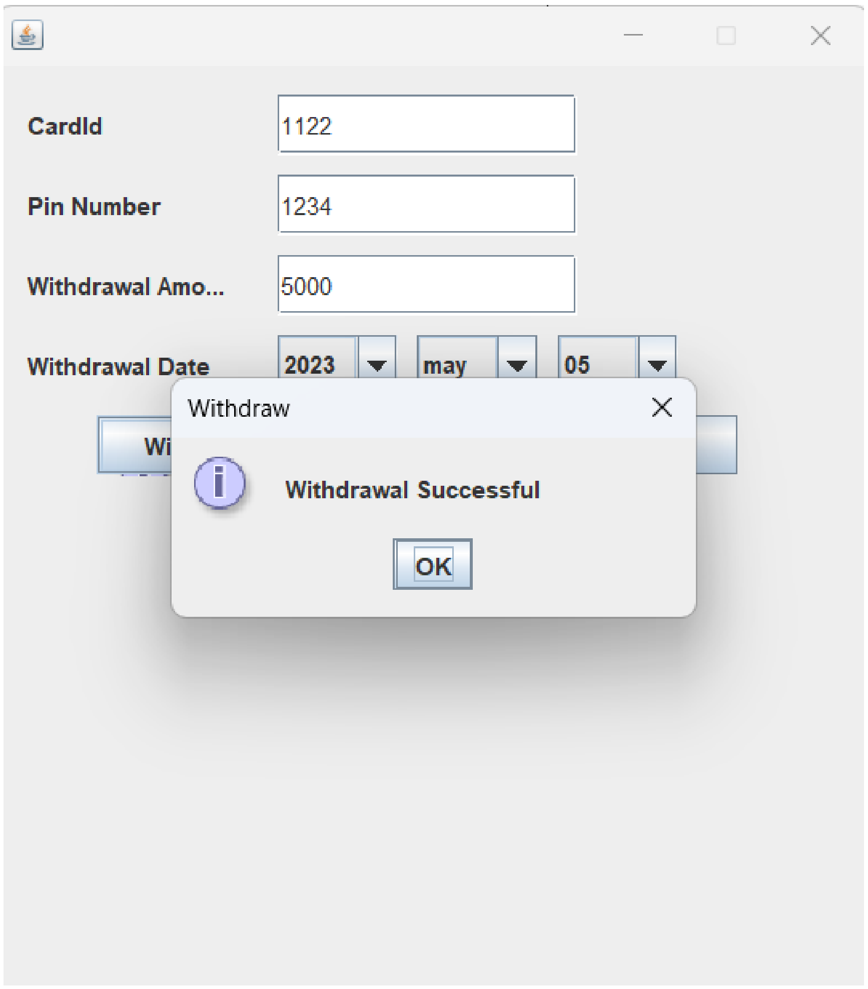
CardId 1122

Pin Number 1234

Withdrawal Amo... 5000

Withdrawal Date 2023 may 05

Figure 10 Inserting Values in Withdrawal Fields


CardId 1122

Pin Number 1234

Withdrawal Amo... 5000

Withdrawal Date 2023 may 05

Withdraw

Withdrawal Successful

Figure 11 After clicking the Withdrawal button.

(d) SET Credit Limit

Table 5 table of setting credit limit

Objective	**SET**ting the credit Limit in Credit Card Class
Action	edit Limit = 20000 ace Period = 12
Expected Result	Credit Limit Has been **SET** dialogue box should be seen.
Actual Result	Credit limit Has been **SET** dialogue box was shown.
Conclusion	The test was successful.

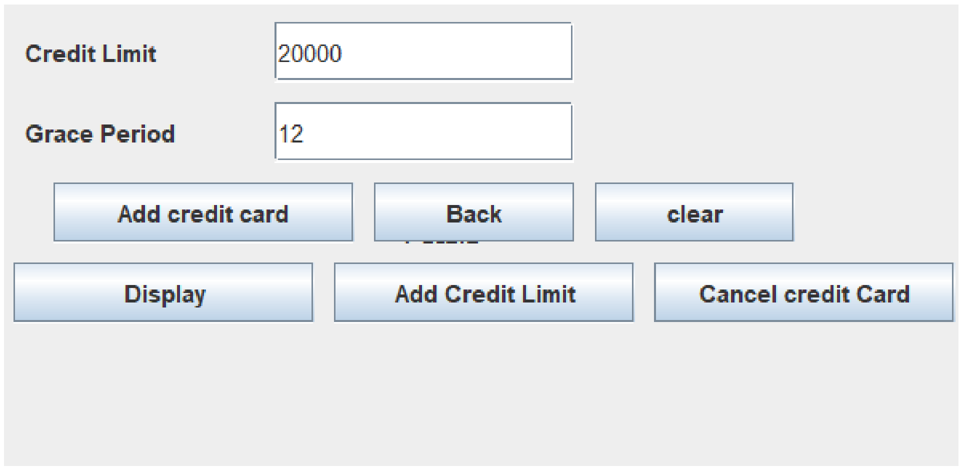
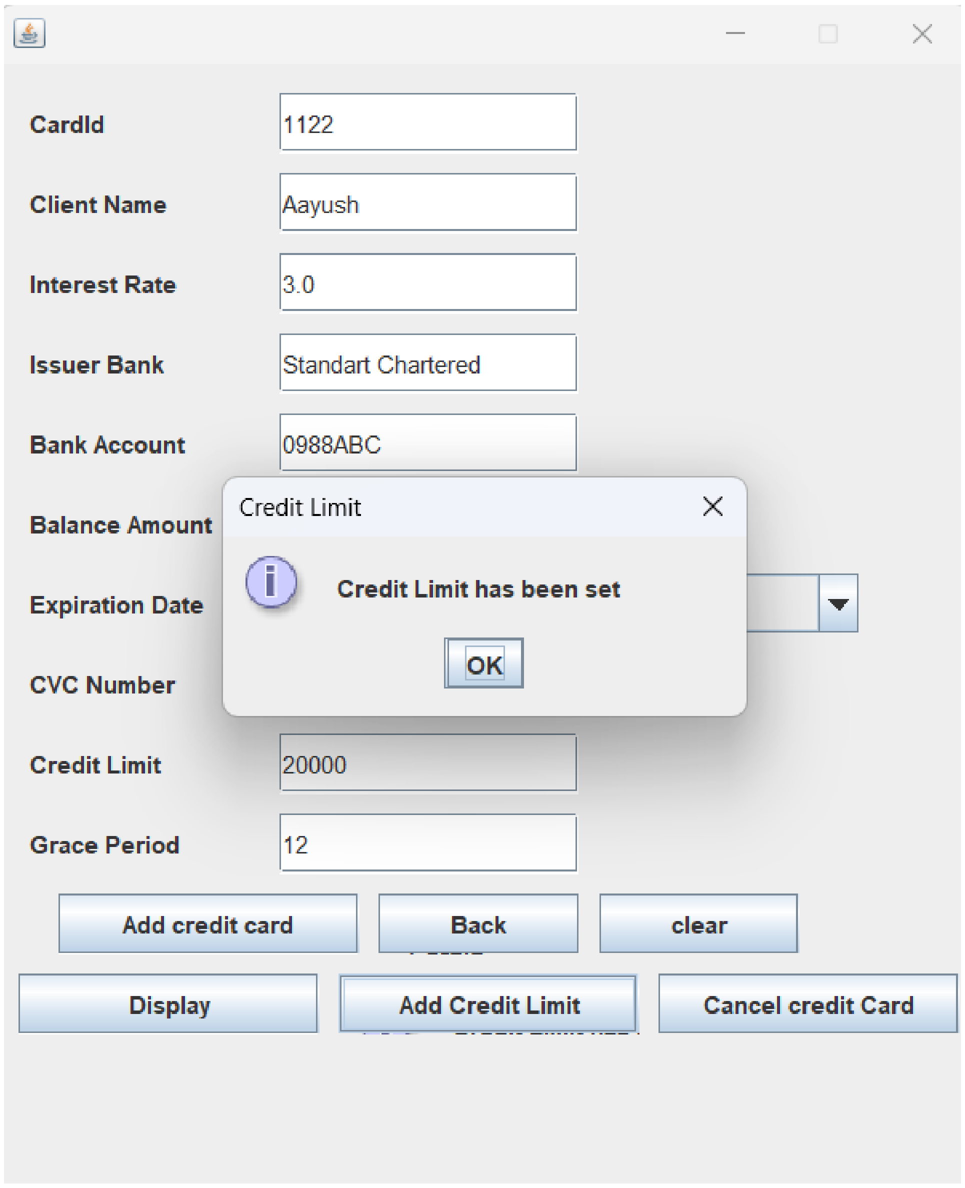

Credit Limit: 20000
Grace Period: 12

Figure 12 Adding Credit Limit in Credit Card Class



CardId	1122
Client Name	Aayush
Interest Rate	3.0
Issuer Bank	Standart Chartered
Bank Account	0988ABC
Balance Amount	
Expiration Date	
CVC Number	
Credit Limit	20000
Grace Period	12

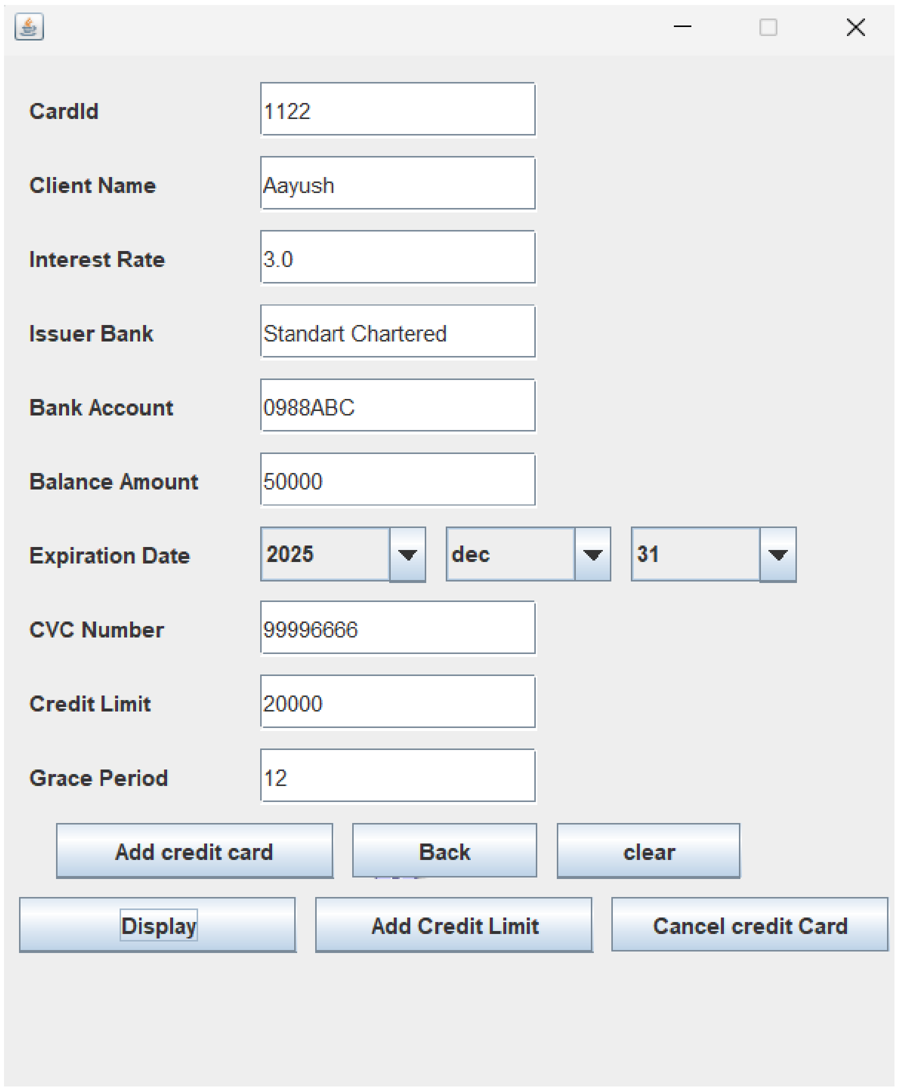
Credit Limit has been set

Figure 13 After Clicking ADD Credit Limit Button

(e) Remove the Credit Card

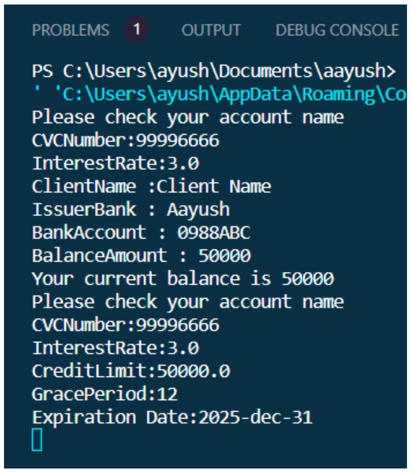
Table 6 table of removing credit card

Objective	Removing the Details of Credit Card Class from array list.
Action	When we press the Cancel Credit Card button the values below should be cleared* while a popup will appear saying Cancelled credit card and if we press display button it will say “your card has been cancelled” Card ID = 1122 Client Name = Aayush Interest Rate = 3.0 Issuer Bank = Standard Chartered Bank Account = 0988ABC Balance Amount = 50000 Expiration Date = 2025 dec 31 CVC Number = 99996666
Expected Result	The credit card must be cancelled.
Actual Result	When we clicked the cancel credit card button it showed us a popup saying Cancelled Credit Card and the card was cancelled.
Conclusion	This test was a success.



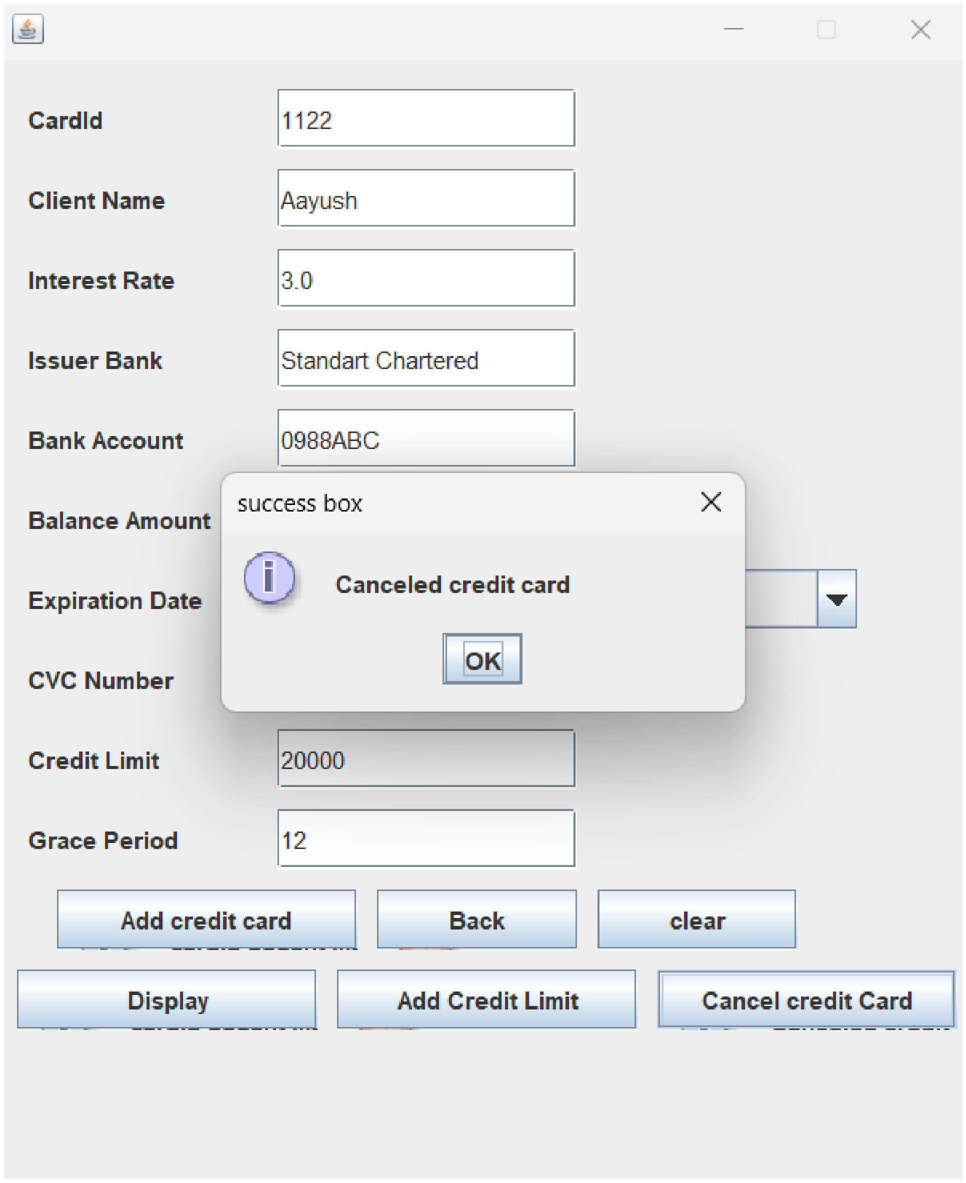
CardId	1122
Client Name	Aayush
Interest Rate	3.0
Issuer Bank	Standart Chartered
Bank Account	0988ABC
Balance Amount	50000
Expiration Date	2025 dec 31
CVC Number	99996666
Credit Limit	20000
Grace Period	12

Figure 14 Value Before Cancelation of Credit Card



```
PS C:\Users\ayush\Documents\aayush>
'  'C:\Users\ayush\AppData\Roaming\Co
Please check your account name
CVCNumber:99996666
InterestRate:3.0
ClientName :Client Name
IssuerBank : Aayush
BankAccount : 0988ABC
BalanceAmount : 50000
Your current balance is 50000
Please check your account name
CVCNumber:99996666
InterestRate:3.0
CreditLimit:50000.0
GracePeriod:12
Expiration Date:2025-dec-31
```

Figure 15 Before Credit Cancelation 2

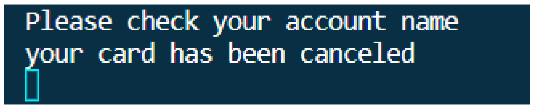


CardId	1122
Client Name	Aayush
Interest Rate	3.0
Issuer Bank	Standart Chartered
Bank Account	0988ABC
Balance Amount	
Expiration Date	
CVC Number	
Credit Limit	20000
Grace Period	12

success box

Canceled credit card

Figure 16 When Pressing Cancel Credit Card Button



```
Please check your account name
your card has been canceled
```

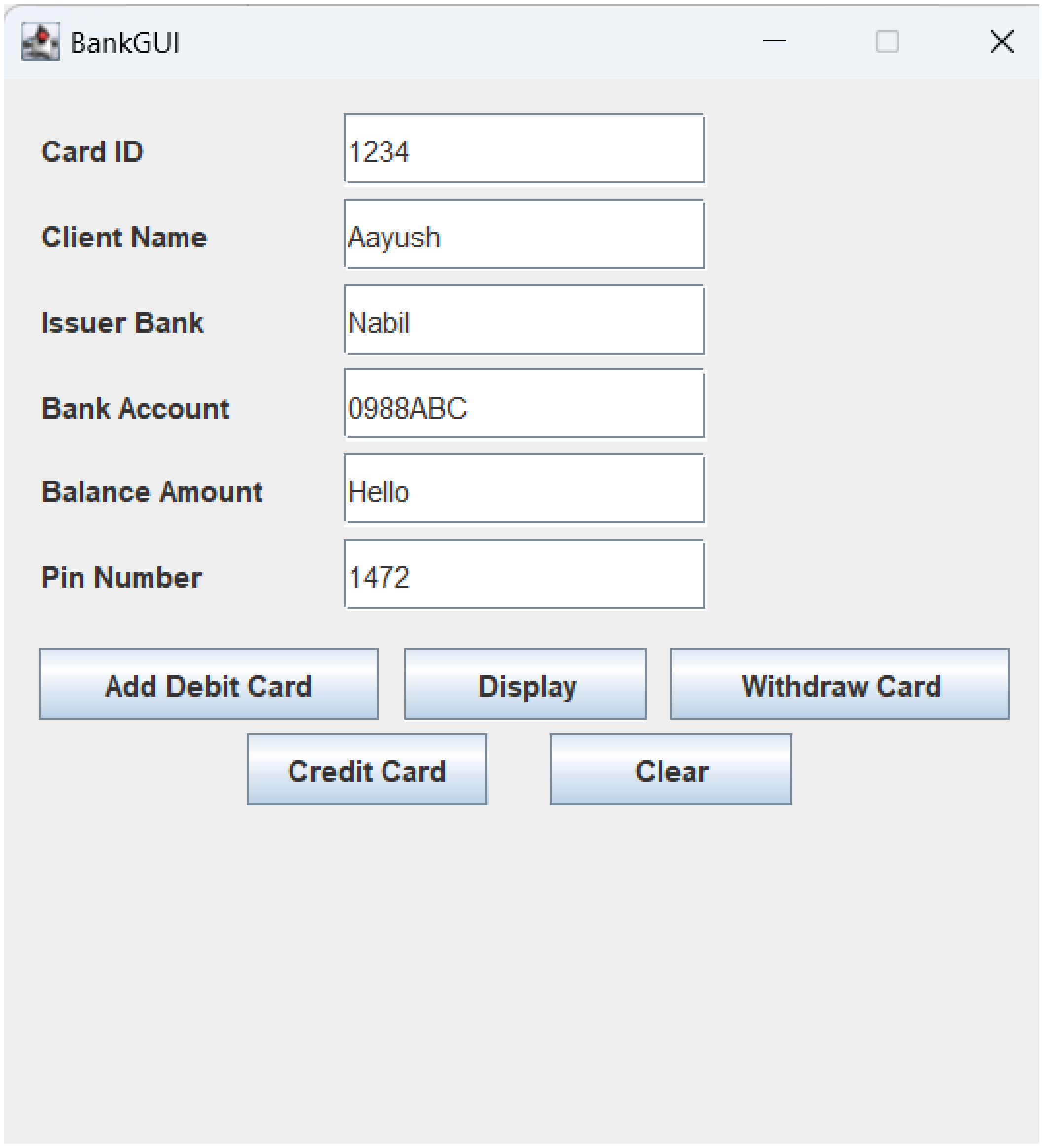
Figure 17 Canceling Credit Card Final Result

5.3 Test 3

5.3.1 Testing dialogue boxes that appear when we input String in a integer field.

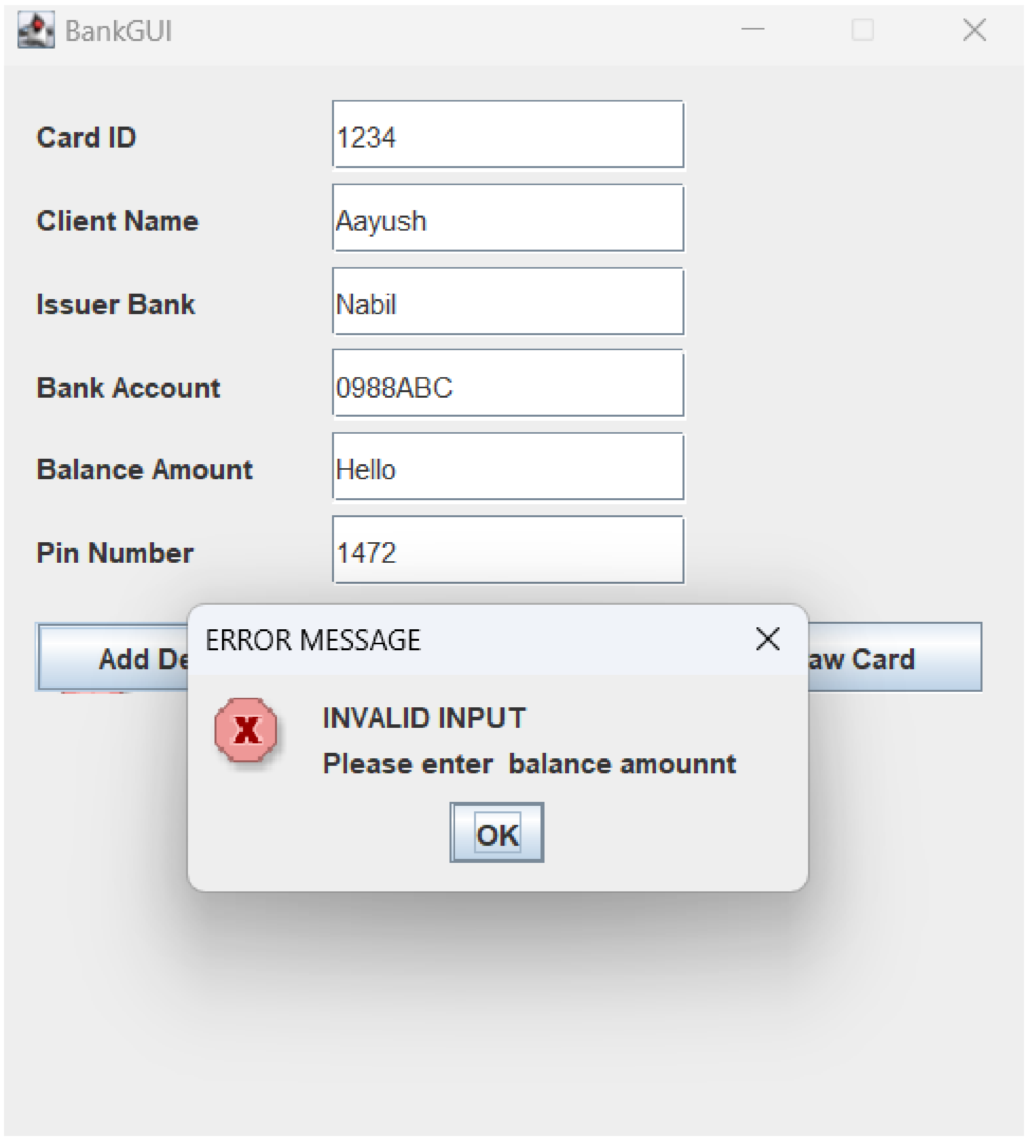
Table 7 table of testing 5.3.1

Objective	To see what the output will be when we input a string in a integer text field.
Action	The hollowing text fields were filled: Card ID = 1234 Client Name: = Aayush Issuer Bank = Standard Chartered BankAccount = 0988ABC Balance Amount = “Hello” Pin Number = 1472 1472Here Balance amount takes in integer data type but instead we inputted “Hello” as input.
Expected result	It should show the message box saying INVALID INPUT Please Input a Valid Balance Amount
Actual result	The popup was shown as expected.
Conclusion	The textbox functions as expected. The test is a success.



Card ID: 1234
Client Name: Aayush
Issuer Bank: Nabil
Bank Account: 0988ABC
Balance Amount: Hello
Pin Number: 1472

Figure 18 Before ADDing String in Integer text field.


BankGUI

Card ID 1234

Client Name Aayush

Issuer Bank Nabil

Bank Account 0988ABC

Balance Amount Hello

Pin Number 1472

ERROR MESSAGE

INVALID INPUT

Please enter balance amounnt

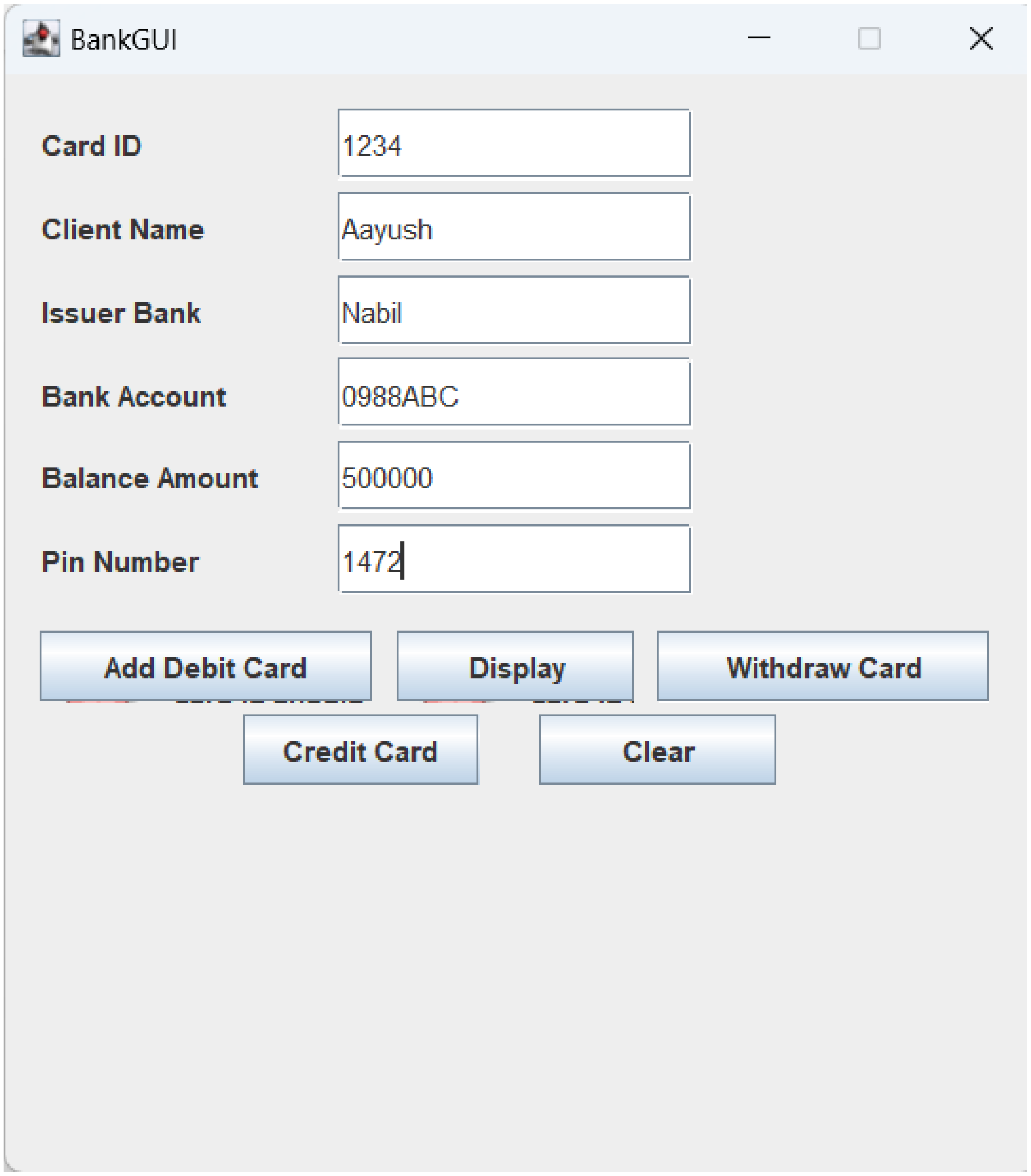
Figure 19 ***After Adding String to an Integer Text field.***

5.3.2 Testing Dialogue box when we enter wrong pin while withdrawing.

Table 8 table of testing 5.3.2

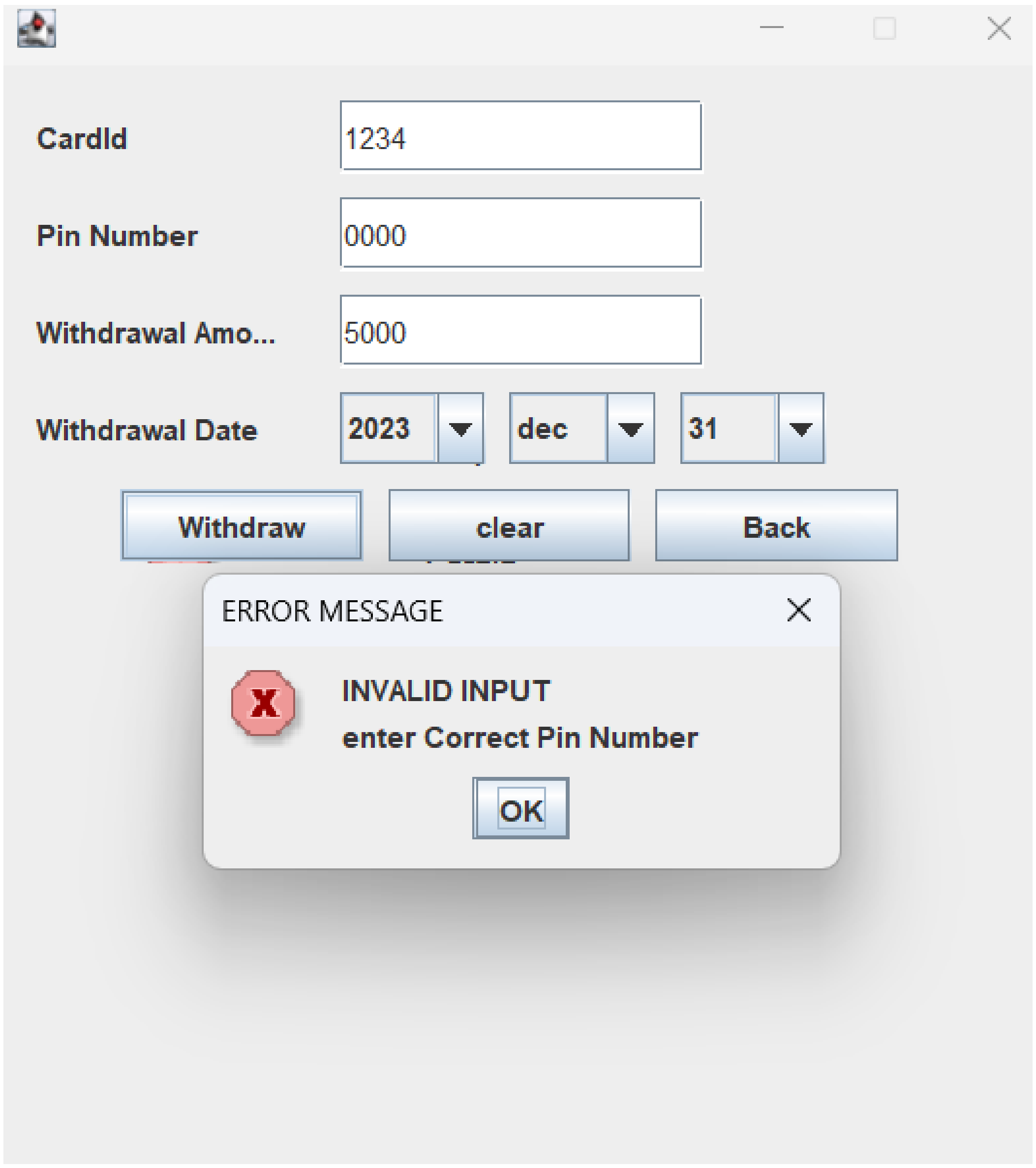
Objective	To see Dialogue box when we enter wrong pin while withdrawing
Action	Fill the details in the Debit Card frame. Card ID = 1234 Client Name = Aayush Issuer Bank = Nabil Bank Account = 0988ABC Balance Amount = 50000 Pin Number = 1472 Withdrawing using incorrect pin Card ID = 1234 Pin Number = 0000 Withdrawal Amount = 5000 Withdrawal Date = 2023-dec-31

Expected Result	Invalid Input Please enter correct pin number dialogue box should be shown.
Actual Result	A dialogue box containing Invalid Input Please enter correct pin number.
Conclusion	It stops the customers from withdrawing when they have entered the wrong pin number. Text was successful.



Card ID	1234
Client Name	Aayush
Issuer Bank	Nabil
Bank Account	0988ABC
Balance Amount	500000
Pin Number	1472

Figure 20 Adding Debit Card values.



CardId 1234

Pin Number 0000

Withdrawal Amo... 5000

Withdrawal Date 2023 dec 31

ERROR MESSAGE

INVALID INPUT
enter Correct Pin Number

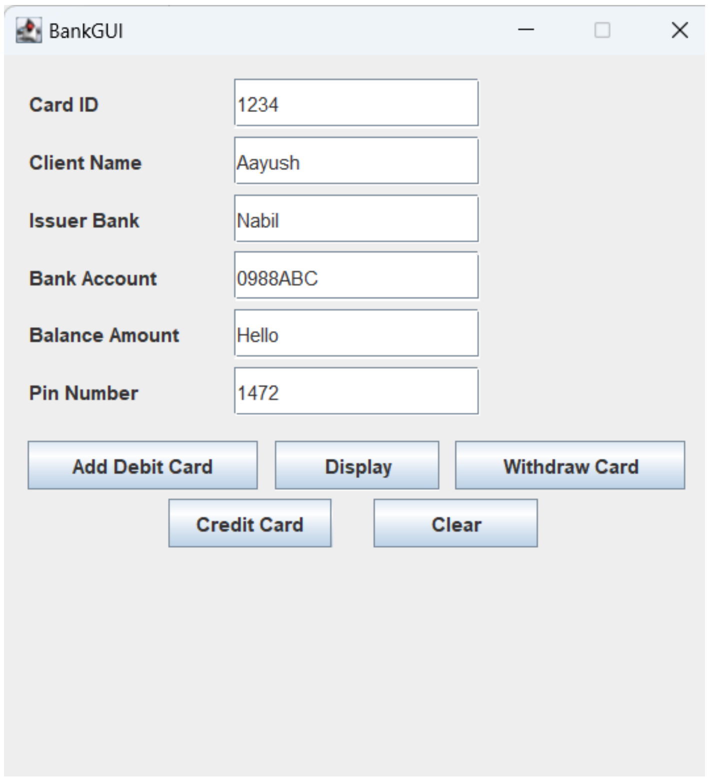
Figure 21 Trying to withdraw using wrong pin.

5.3.3 Trying to register two accounts in debit card having similar Card ID

Table 9table of test of 5.3.3

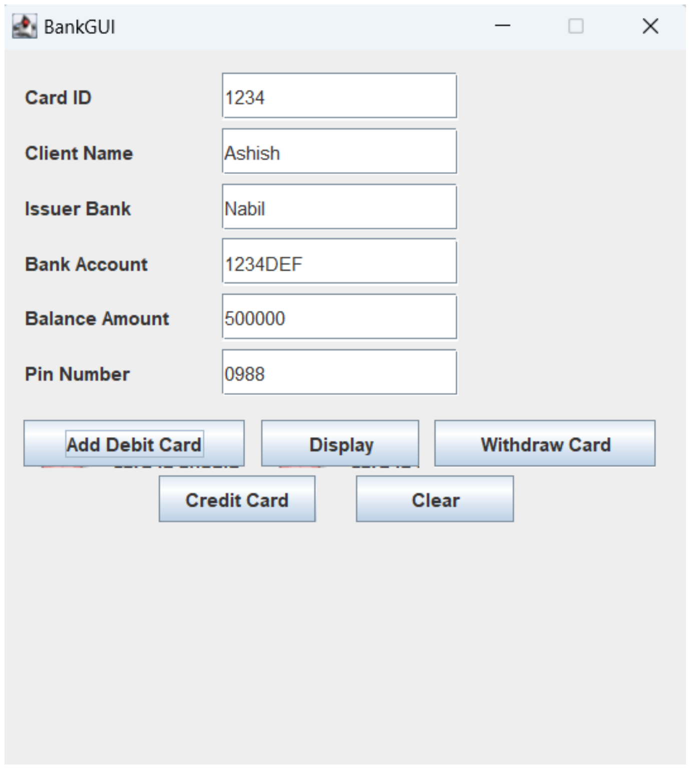
Objective	Trying to register two accounts in debit card having similar Card ID
Action	Registering two debit card accounts with same pin number: Account no 1: Card ID = 1234 Client Name = Aayush Issuer Bank = Nabil Bank Account = 0988ABC Balance Amount = 50000 Pin Number = 1472 Account no 2: Card ID = 1234 Client Name = Ashish Issuer Bank = Nabil Bank Account = 0988ABC Balance Amount = 50000 Pin Number = 0988 Here both accounts have the same Card ID.
Expected Result	

	Registration of number 2 account should not be possible. A dialogue box should popup saying card id must be unique.
Actual Result	Registration of account number 2 was stopped.
Conclusion	The test was successful.



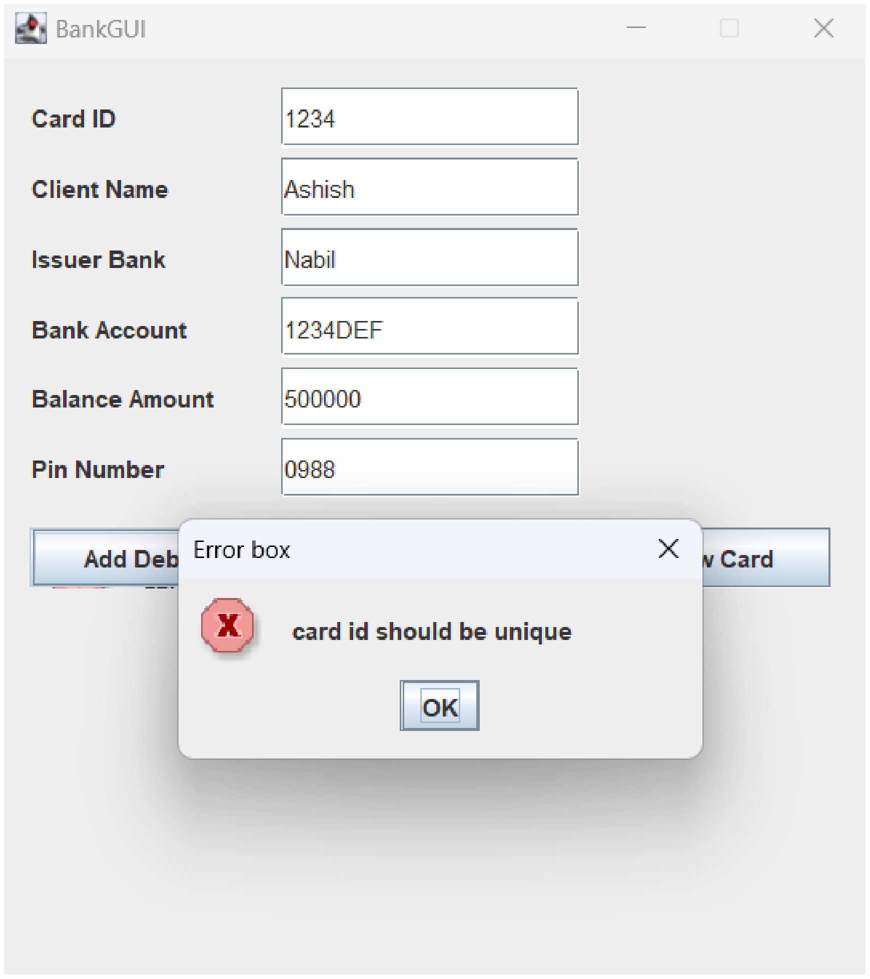
Card ID	1234
Client Name	Aayush
Issuer Bank	Nabil
Bank Account	0988ABC
Balance Amount	Hello
Pin Number	1472

Figure 22 Account number 1 with same Card ID



Card ID	1234
Client Name	Ashish
Issuer Bank	Nabil
Bank Account	1234DEF
Balance Amount	500000
Pin Number	0988

Figure 23 Account Number 2 with same pin number



Card ID: 1234
Client Name: Ashish
Issuer Bank: Nabil
Bank Account: 1234DEF
Balance Amount: 500000
Pin Number: 0988

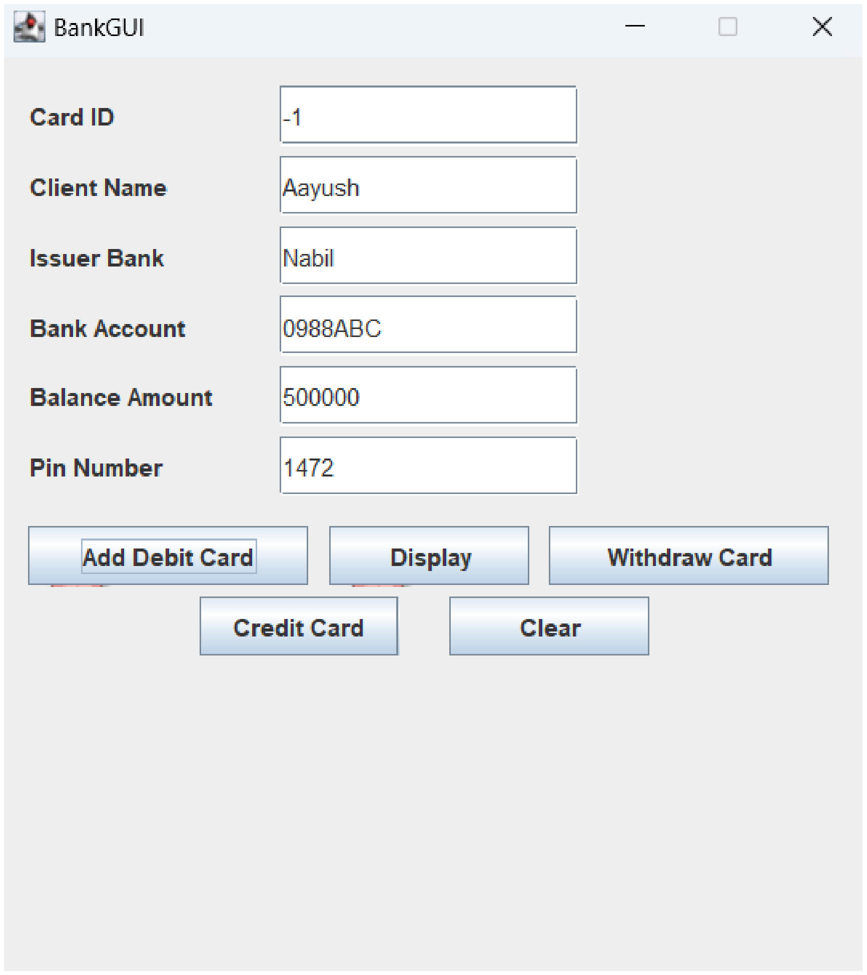
card id should be unique

Figure 24 Cancelation of registration of account number 2

5.3.4 Testing dialogue box that appears when we input negative cardId.

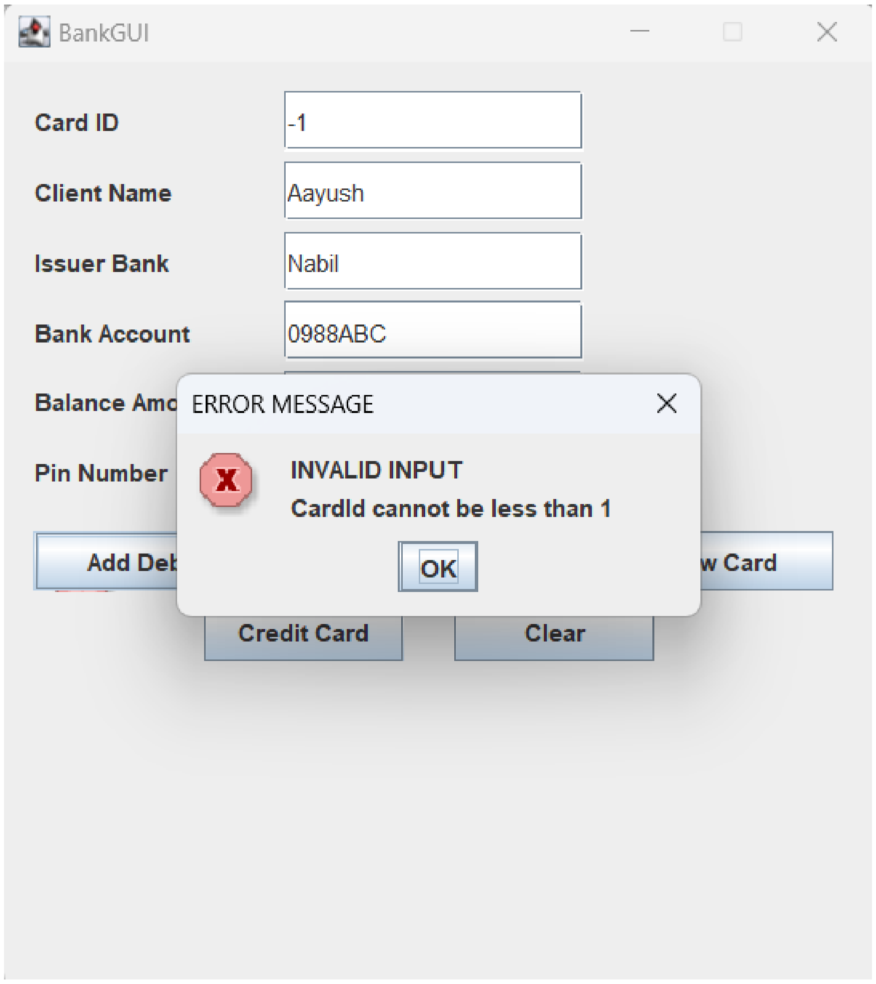
Table 10table of test of 5.3.4

Objective	Testing dialogue box that appears when we input negative cardId
Action	Registration of user with negative card id: Card ID = -1 Client Name = Aayush Issuer Bank = Nabil Bank Account = 0988ABC Balance Amount = 50000 Pin Number = 1472
Expected Results	Registration should not be done and an error message dialog box need to appear
Actual Results	The dialog box containing the error message "Invalid i Input card id cannot be less than 1.
Conclusion	When negative data is entered* dialog windows function. The test is a success.



Card ID: -1
Client Name: Aayush
Issuer Bank: Nabil
Bank Account: 0988ABC
Balance Amount: 500000
Pin Number: 1472

Figure 25 Debit Card with Negative Card ID



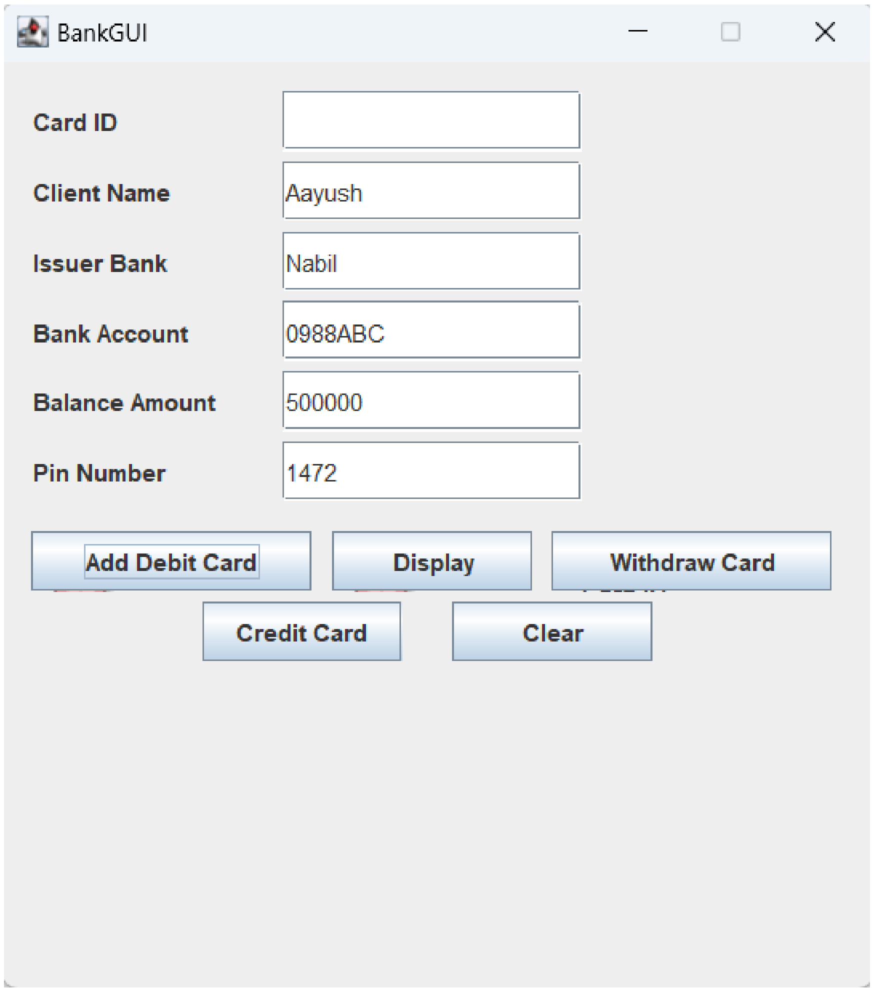
INVALID INPUT
CardId cannot be less than 1

Figure 26 Trying to Register card id which has negative value.

5.3.5 Testing dialog boxes that appear when text fields are empty.

Table 11table of test of 5.3.5

Objective	to see if the right dialog box shows or not when all of the text fields are not filled in.
Action	Inputting Value in debit Card: Card ID = Client Name = Aayush Issuer Bank = Nabil Bank Account = 0988ABC Balance Amount = 50000 Pin Number = 1472
Expected Results	An error message dialog box needs to appear.
Actual Results	The error notice "Please fill out all the details" appears in a dialog box.
Conclusion	Messages appear in dialog windows when text fields are left blank. The test was a success.



Card ID	
Client Name	Aayush
Issuer Bank	Nabil
Bank Account	0988ABC
Balance Amount	500000
Pin Number	1472

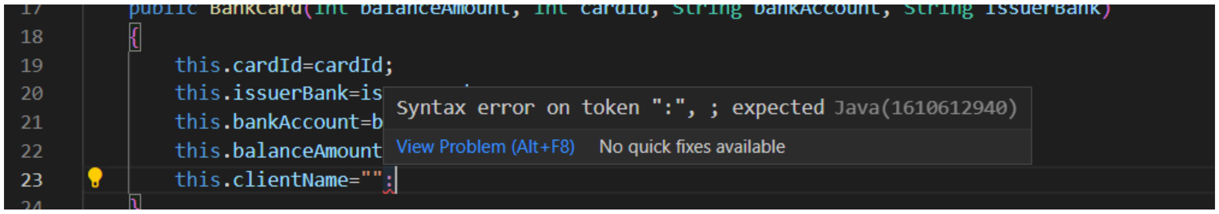
Figure 27 Before Adding Debit Card with empty Card ID

6. Error and Detection

6.1 Syntax Error

When a program is constructed poorly or violates any syntax rules, it can cause syntax errors. Every computer program must follow strict syntax to compile and run properly. Therefore, when a program is constructed poorly or violates any syntax rules, it can cause syntax errors.

Here is a screenshot of a syntax error that was found during debugging. After a few tries of unsuccessful compilation, I found that there was a colon instead of semicolon in line 23. This prevented the program from compiling and run.

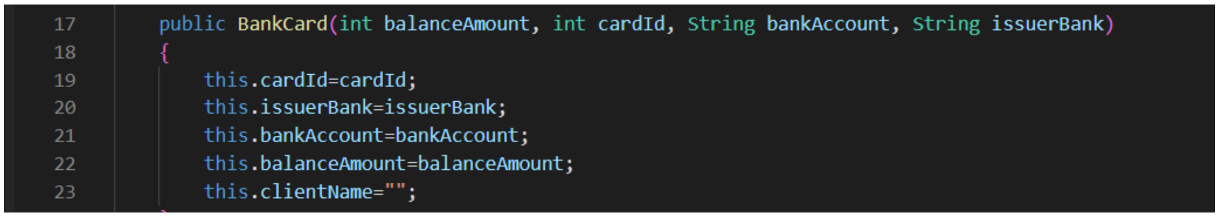


```
17      public BankCard(int balanceAmount, int cardId, String bankAccount, String issuerBank)
18      {
19          this.cardId=cardId;
20          this.issuerBank=is
21          this.bankAccount=b
22          this.balanceAmount
23          this.clientName="":
```

```
Syntax error on token ":", ; expected Java(1610612940)
View Problem (Alt+F8)    No quick fixes available
```

Figure 28 figure of syntax error

For that, I went through each line and found the error and added a semicolon at the end of line 23. This solved the error* and the programed compile successfully.

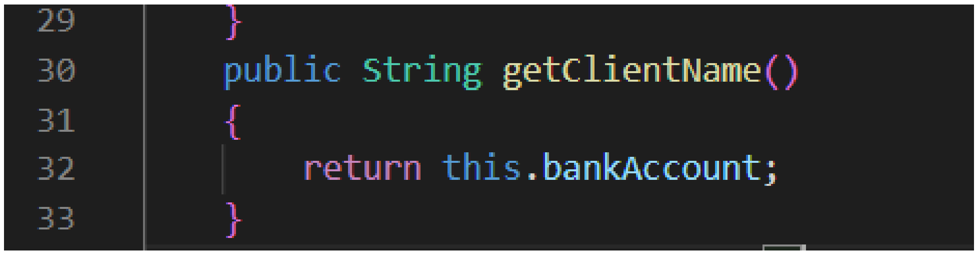


```
17      public BankCard(int balanceAmount, int cardId, String bankAccount, String issuerBank)
18      {
19          this.cardId=cardId;
20          this.issuerBank=issuerBank;
21          this.bankAccount=bankAccount;
22          this.balanceAmount=balanceAmount;
23          this.clientName="";
```

Figure 29 figure of rectifying syntax error

6.2 Semantic error

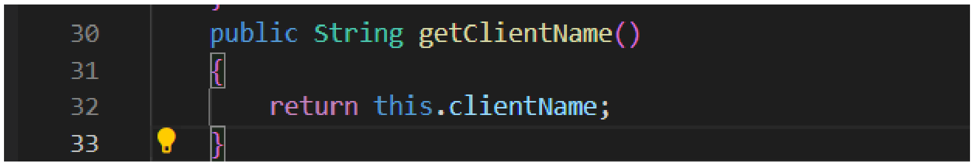
Semantic errors can occur when the incorrect variable or operator is used instead of the one that was intended to be used. While creating a method named getClientName()* I mistakenly wrote a different return type variable which was bankAccount when there supposed to be clientName in line 32. Here is the screenshot of program with wrong return type variable.



```
29        }
30        public String getClientName()
31        {
32            return this.bankAccount;
33        }
```

Figure 30 figure of semantic error

When I encountered this error* I simply corrected return type variable with clientName and this solved the issue. Here is the screenshot of program after changes.



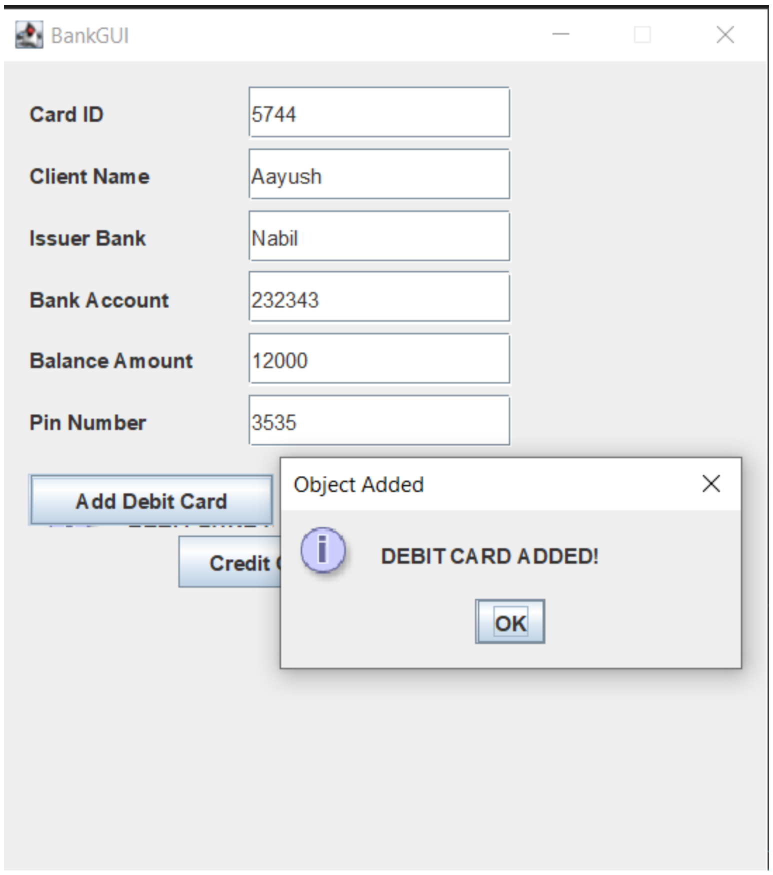
```
30        public String getClientName()
31        {
32            return this.clientName;
33        }
```

Figure 31 figure of rectifying semantic error

6.3 Logical Error

A computer program must be logically correct to function. For any program that does not have a clear logic of what it should do may cause trouble resulting in a different result than expected. To put it simply, logical error is a kind of programming error when result of the program is different from the instruction that are given to it.

During the development of this program, I came across few logical errors mainly while working on Bank Card. After adding debit card* with input values as shown in the figure:



BankGUI

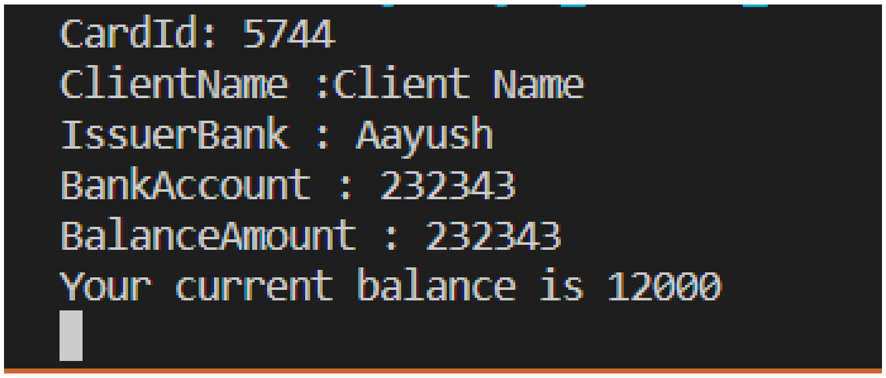
Card ID	5744
Client Name	Aayush
Issuer Bank	Nabil
Bank Account	232343
Balance Amount	12000
Pin Number	3535

Object Added

DEBIT CARD ADDED!

Figure 32 figure of logical error

In the above picture I have added 232343 and 12000 as bank account and balance respectively. But when I tried to display the details after the addition, I received the same value of bank account and balance amount.

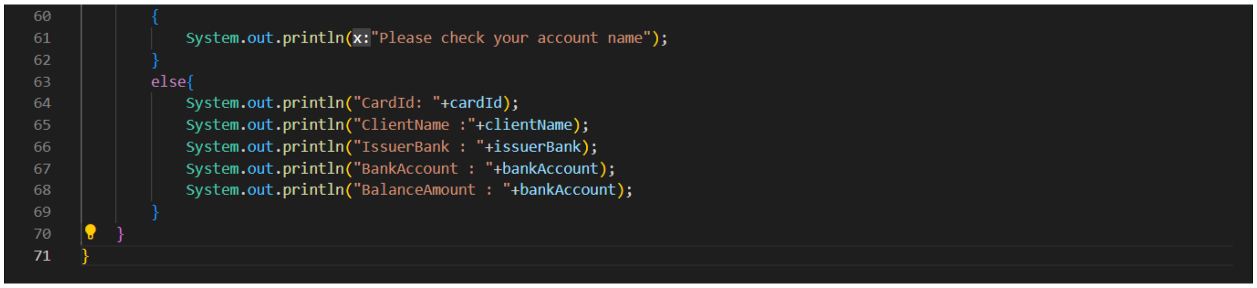


```
CardId: 5744
ClientName :Client Name
IssuerBank : Aayush
BankAccount : 232343
BalanceAmount : 232343
Your current balance is 12000
```

Figure 33 figure of logical error(2)

Here the expected output was bank account 232343 and balance amount 12000 but as I have called the bank account instead of balance amount* I got the wrong value.

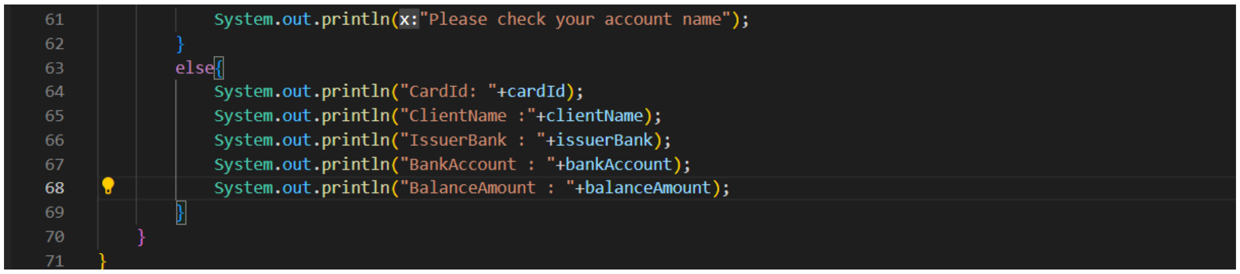
When I looked for the error I found that I have written bank account in line 68. There is supposed to be balanceAmount instead of bankAccount. This gave the program to display the same value as bankAccount in the place of bank Amount as well. This prevented the program from displaying the correct value.



```
60              {
61                  System.out.println(x:"Please check your account name");
62              }
63              else{
64                  System.out.println("CardId: "+cardId);
65                  System.out.println("ClientName :"+clientName);
66                  System.out.println("IssuerBank : "+issuerBank);
67                  System.out.println("BankAccount : "+bankAccount);
68                  System.out.println("BalanceAmount : "+bankAccount);
69              }
70          }
71      }
```

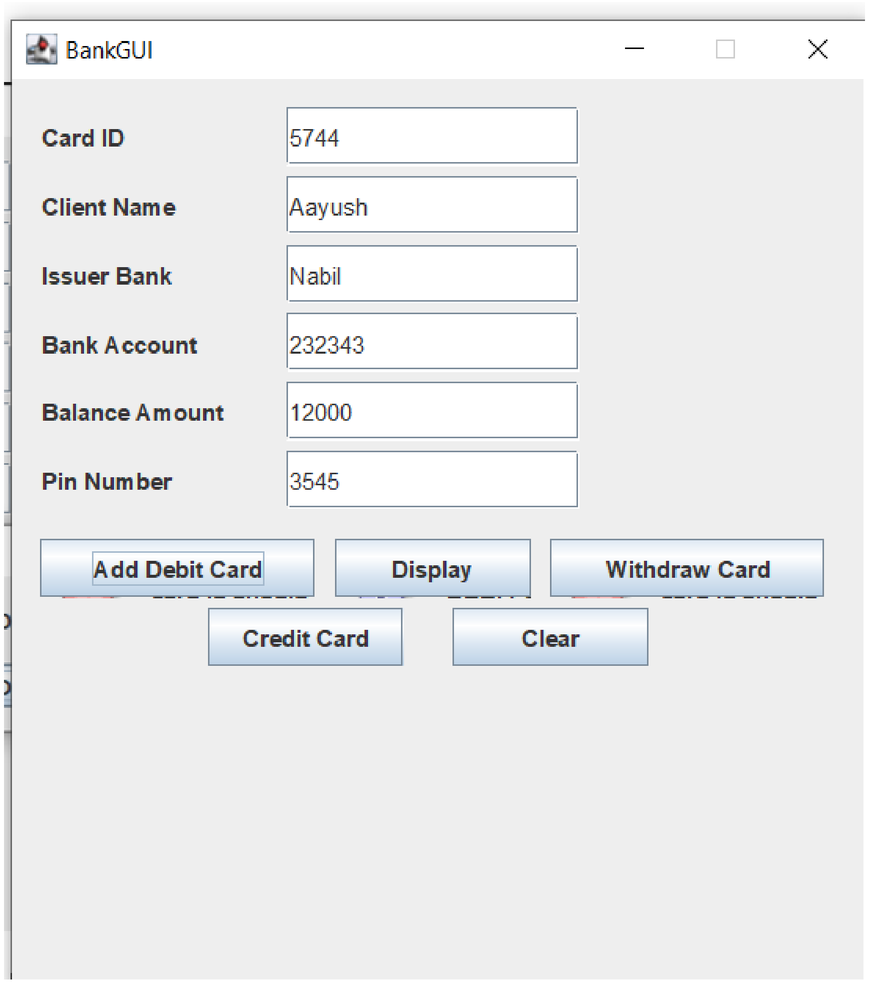
Figure 34 figure of rectyfying logical error

To solve this error I changed the bankAccount to the balance amount to get the correct value in line 68. The error was solved, and I got the result I was expecting.



```
                System.out.println(x:"Please check your account name");
            }
            else{
                System.out.println("CardId: "+cardId);
                System.out.println("ClientName :"+clientName);
                System.out.println("IssuerBank : "+issuerBank);
                System.out.println("BankAccount : "+bankAccount);
                System.out.println("BalanceAmount : "+balanceAmount);
            }
        }
}
```

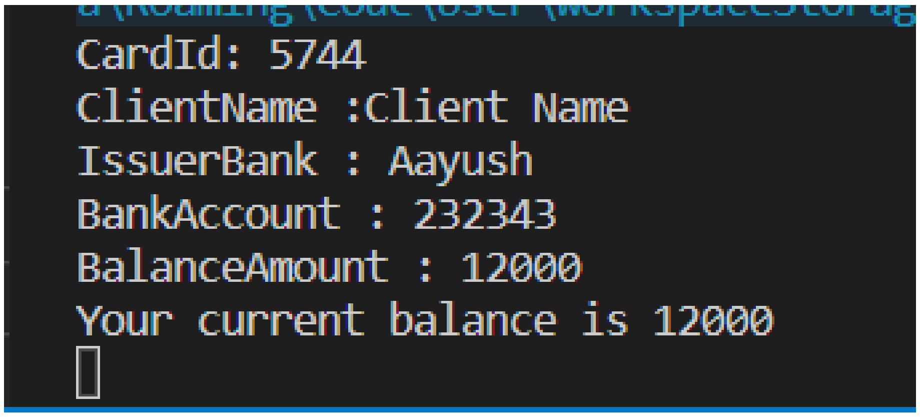
Figure 35 figure of rectifying logical error(2)



BankGUI

Card ID	5744
Client Name	Aayush
Issuer Bank	Nabil
Bank Account	232343
Balance Amount	12000
Pin Number	3545

Figure 36 figure of rectifying logical error(3)



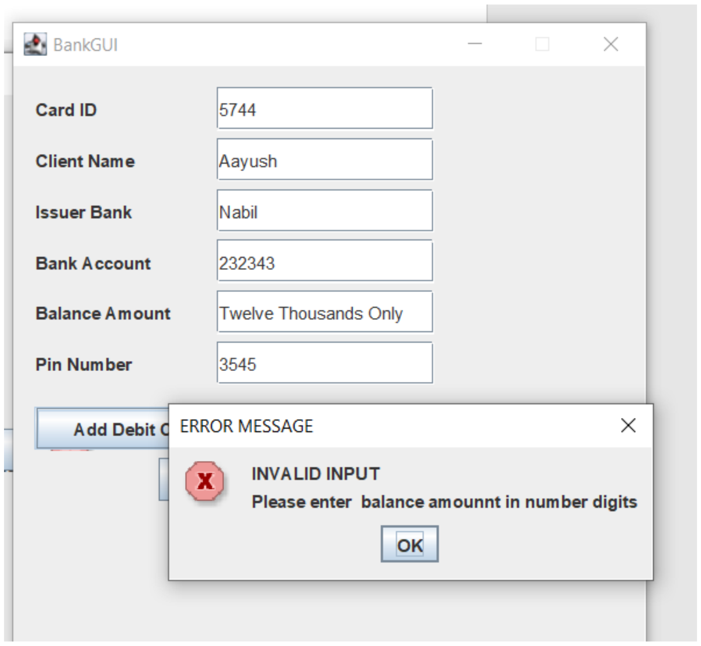
```
CardId: 5744
ClientName :Client Name
IssuerBank : Aayush
BankAccount : 232343
BalanceAmount : 12000
Your current balance is 12000
```

Figure 37 figure of logical error (4)

6.4 Run-time Error

Run-time errors occur only when the program runs. We can get this error even when our program is syntactically correct and compiles without any issues.

Here is a screenshot of a Run-time error that I have found while debugging the program. I have given the string values in balance amount whereas the program was instructed to get them in number/digits value. This gave an error message asking the user to give input in numbers.



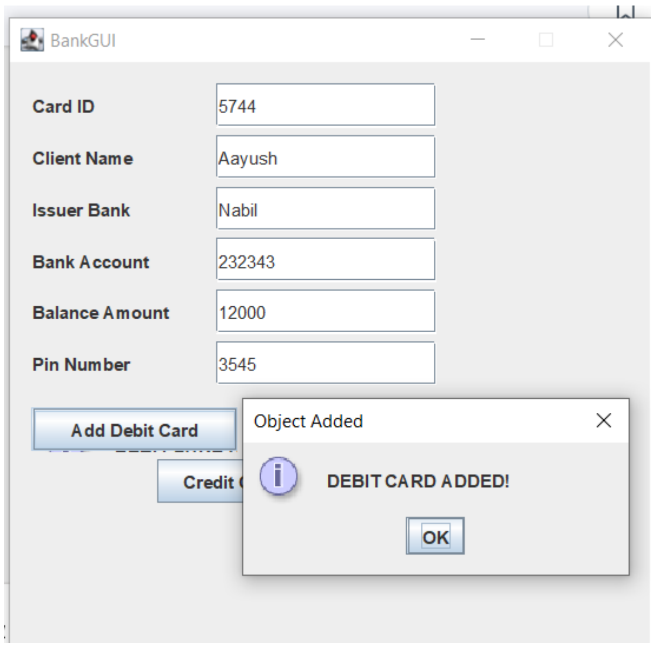
BankGUI

Card ID: 5744
Client Name: Aayush
Issuer Bank: Nabil
Bank Account: 232343
Balance Amount: Twelve Thousands Only
Pin Number: 3545

ERROR MESSAGE
INVALID INPUT
Please enter balance amounnt in number digits

Figure 38 figure of run time error

Below is the screenshot of the program accepting the card when the balance amount is given in numbers.



BankGUI

Card ID: 5744
Client Name: Aayush
Issuer Bank: Nabil
Bank Account: 232343
Balance Amount: 12000
Pin Number: 3545

Object Added
DEBIT CARD ADDED!

Figure 39 figure of rectifying run time error

7. Conclusion

Completing this coursework was a real challenge yet a new experience that helped me enhance my learning in Java. I had the opportunity to learn different aspects of Java error handling, methods and mainly GUI. At first, I really was overwhelmed by the scenarios in the coursework. I researched and went through every module resource provided by our tutors. I begin with the question itself; I seek help from my module tutors and discussed it with my friends in class. After understanding the requirements of coursework I begin with the planning and designing GUI.

While working on methods I came across several issues and problems which thankfully were solved with proper research and with the help of our tutor and my classmates. GUI was fun as well as tiring I would say. The fun part was playing with the components and sizes. I got to learn setBounds methods for the size and positions, labels, frame and many more.

This course has helped me to broaden my understanding of Java. Each line has its own meaning, and working on the concept of inheritance has been a great experience. The overall coursework has taught me not only to complete the given task but to think outside the box and handle the possible errors and exceptions that might occur in the life of a programmer.

Lastly, I would like to sum up by showing my gratitude towards my module teachers who have been guiding me throughout the entire journey and my classmates for helping me to understand the simplest confusions and problems. The best experience besides my goal to complete the work is to focus on consistency. At the time I felt like I could not do it I gained the motivation from the loved ones around me which helped me to put consistency on my work and as a result I am now able to complete my task on time. I still am looking forward to learning more about programming in future as well.

8. References

Oracle. (2023, March 21). *ORACLE.* Retrieved from ORACLE: https://www.oracle.com/java/technologies/javase/jdk-jdk-7-readme.html

Red Hat. (2020, April 22). *JRE.* Retrieved from RedHat: https://www.redhat.com/en/topics/cloud-native-apps/what-is-a-Java-runtime-environment

Tyson, M. (2022, October 28). *JVM*. Retrieved from InfoWorld: https://www.infoworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html

9. Appendix

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
public class BankGUI
{
   private JFrame frame1, frame2, frame3;
       private JLabel cardId,clientName,issuerBank,bankAccount,balanceAmount,
pinNumber, withdrawalAmount, withdrawalDate, CVCNumber, creditLimit, InterestRate,
graceperiod, expirationDate, withdrawcardId, withdrawpinNumber, creditcardId,
creditClientName,creditIssuerBank, creditBankAccount, creditBalanceAmount;
       private JTextField textField1, textField2, textField3, textField4, textField5,
textField6, textField7, textField8, textField9, textField10, textField11, textField12,
textField13, textField14, textField15, textField16, textField17, textField18;
       private JButton adddebit, DisplayD, creditcard, clear,withdrawcard, withdraw,
withdarwclear,withdrawback, back, addcredit, creditclear, addcreditLimit, cancelcredit,
Displayc;
       private JComboBox<String> years, months, days, years2, months2, days2;
       private  String[] year=
{"year","2020","2021","2022","2023","2024","2025","2026","2027","2028","2029","2030","
2031","2032","2033"};
       private String[] month= {"month","jan", "feb", "mar", "apr", "may", "jun", "jul", "sep",
"oct", "nov", "dec"};
       private String[] day= {"day","01","02", "03" ,"04","05" ,"06" ,"07" ,"08" ,"09","10"
,"10"
```

```
,"11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","2
9","30","31"};
        private final static int INVALID = -1;
        ArrayList<BankCard> array = new ArrayList<BankCard>();
    public BankGUI()
    {
        frame1 = new JFrame("BankGUI");
        frame1.setSize(450, 500);
        frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame1.setLayout(null);
        frame1.setResizable(false);
        frame1.setLocationRelativeTo(null);

        cardId = new JLabel("Card ID");
        cardId.setBounds(15, 15, 100, 30);
        frame1.add(cardId);
        textField1 = new JTextField();
        textField1.setBounds(140, 15, 150, 30);
        frame1.add(textField1);

        clientName = new JLabel("Client Name");
        clientName.setBounds(15, 50, 100, 30);
        frame1.add(clientName);
        textField2 = new JTextField();
        textField2.setBounds(140, 50, 150, 30);
        frame1.add(textField2);

        issuerBank = new JLabel("Issuer Bank");
        issuerBank.setBounds(15, 85, 100, 30);
```

```
frame1.add(issuerBank);
textField3 = new JTextField();
textField3.setBounds(140, 85, 150, 30);
frame1.add(textField3);

bankAccount = new JLabel("Bank Account");
bankAccount.setBounds(15, 120, 100, 30);
frame1.add(bankAccount);
textField4 = new JTextField();
textField4.setBounds(140, 120, 150, 30);
frame1.add(textField4);

balanceAmount = new JLabel("Balance Amount");
balanceAmount.setBounds(15, 155, 100, 30);
frame1.add(balanceAmount);
textField5 = new JTextField();
textField5.setBounds(140, 155, 150, 30);
frame1.add(textField5);

pinNumber = new JLabel("Pin Number");
pinNumber.setBounds(15, 190, 100, 30);
frame1.add(pinNumber);
textField6 = new JTextField();
textField6.setBounds(140, 190, 150, 30);
frame1.add(textField6);

adddebit = new JButton("Add Debit Card");
adddebit.setBounds(15, 235, 140, 30);
```

```
frame1.add(adddebit);
adddebit.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        addDebit();
    }
});


DisplayD = new JButton("Display");
DisplayD.setBounds(165, 235, 100, 30);
frame1.add(DisplayD);
DisplayD.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        showDebit();
    }
});

withdrawcard = new JButton("Withdraw Card");
withdrawcard.setBounds( 275, 235, 140, 30);
frame1.add(withdrawcard);
withdrawcard.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        frame2.setVisible(true);
        frame1.dispose();
    }
});

creditcard = new JButton("Credit Card");
creditcard.setBounds( 100, 270, 100, 30);
frame1.add(creditcard);
creditcard.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        frame3.setVisible(true);
        frame1.dispose();

    }
});

clear = new JButton("Clear");
clear.setBounds( 225, 270, 100, 30);
frame1.add(clear);
clear.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        textField1.setText("");
        textField2.setText("");
        textField3.setText("");
```

```
                textField4.setText("");
                textField5.setText("");
                textField6.setText("");
            }
        });


        frame1.setVisible(true);



        frame2 = new JFrame();
        frame2.setSize(450, 500);
        frame2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame2.setLayout(null);
        frame2.setResizable(false);
        frame2.setLocationRelativeTo(null);

        withdrawcardId = new JLabel("CardId");
        withdrawcardId.setBounds(15, 15, 100, 30);
        frame2.add(withdrawcardId);
        textField7 = new JTextField();
        textField7.setBounds(140, 15, 150, 30);
        frame2.add(textField7);

        withdrawpinNumber = new JLabel("Pin Number");
        withdrawpinNumber.setBounds(15, 55, 100, 30);
        frame2.add(withdrawpinNumber);
        textField8 = new JTextField();
```

```
textField8.setBounds(140, 55, 150, 30);
frame2.add(textField8);

withdrawalAmount = new JLabel("Withdrawal Amount");
withdrawalAmount.setBounds(15, 95, 100, 30);
frame2.add(withdrawalAmount);
textField9 = new JTextField();
textField9.setBounds(140, 95, 150, 30);
frame2.add(textField9);

withdrawalDate = new JLabel("Withdrawal Date");
withdrawalDate.setBounds(15, 135, 100, 30);
frame2.add(withdrawalDate);
years = new JComboBox<String>(year);
years.setBounds(140, 135, 60, 30);
frame2.add(years);
months = new JComboBox<String>(month);
months.setBounds(210, 135, 60, 30);
frame2.add(months);
days = new JComboBox<String>(day);
days.setBounds(280, 135, 60, 30);
frame2.add(days);

withdraw = new JButton("Withdraw");
withdraw.setBounds( 50, 175, 100, 30);
frame2.add(withdraw);
withdraw.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        Withdraw();
    }
});

withdarwclear = new JButton("clear");
withdarwclear.setBounds( 160, 175, 100, 30);
frame2.add(withdarwclear);
withdarwclear.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        textField7.setText("");
        textField8.setText("");
        textField9.setText("");
        years.setSelectedIndex(0);
        months.setSelectedIndex(0);
        days.setSelectedIndex(0);
    }
});

withdrawback = new JButton("Back");
withdrawback.setBounds( 270, 175, 100, 30);
frame2.add(withdrawback);
withdrawback.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        frame1.setVisible(true);
        frame2.dispose();
    }
});

frame3 = new JFrame();
frame3.setSize(500, 600);
frame3.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame3.setLayout(null);
frame3.setResizable(false);
frame3.setLocationRelativeTo(null);



creditcardId = new JLabel("CardId");
creditcardId.setBounds(15, 15, 100, 30);
frame3.add(creditcardId);
textField12 = new JTextField();
textField12.setBounds(140, 15, 150, 30);
frame3.add(textField12);

creditClientName = new JLabel("Client Name");
creditClientName.setBounds(15, 55, 100, 30);
frame3.add(creditClientName);
textField15 = new JTextField();
textField15.setBounds(140, 55, 150, 30);
frame3.add(textField15);
```

```
InterestRate = new JLabel("Interest Rate");
InterestRate.setBounds(15, 95, 100, 30);
frame3.add(InterestRate);
textField13 = new JTextField();
textField13.setBounds(140, 95, 150, 30);
frame3.add(textField13);



creditIssuerBank= new JLabel("Issuer Bank");
creditIssuerBank.setBounds(15, 135, 100, 30);
frame3.add(creditIssuerBank);
textField16 = new JTextField();
textField16.setBounds(140, 135, 150, 30);
frame3.add(textField16);

creditBankAccount = new JLabel("Bank Account");
creditBankAccount.setBounds(15, 175, 100, 30);
frame3.add(creditBankAccount);
textField17 = new JTextField();
textField17.setBounds(140, 175, 150, 30);
frame3.add(textField17);

creditBalanceAmount = new JLabel("Balance Amount");
creditBalanceAmount.setBounds(15, 215, 100, 30);
frame3.add(creditBalanceAmount);
textField18 = new JTextField();
```

```
textField18.setBounds(140, 215, 150, 30);
frame3.add(textField18);

expirationDate = new JLabel("Expiration Date");
expirationDate.setBounds(15, 255, 100, 30);
frame3.add(expirationDate);
years2 = new JComboBox<String>(year);
years2.setBounds(140, 255, 90, 30);
frame3.add(years2);
months2 = new JComboBox<String>(month);
months2.setBounds(240, 255, 90, 30);
frame3.add(months2);
days2 = new JComboBox<String>(day);
days2.setBounds(340, 255, 90, 30);
frame3.add(days2);

CVCNumber = new JLabel("CVC Number");
CVCNumber.setBounds(15, 295, 100, 30);
frame3.add(CVCNumber);
textField10 = new JTextField();
textField10.setBounds(140, 295, 150, 30);
frame3.add(textField10);

creditLimit = new JLabel("Credit Limit");
creditLimit.setBounds(15, 335, 100, 30);
frame3.add(creditLimit);
textField11 = new JTextField();
textField11.setBounds(140, 335, 150, 30);
```

```
frame3.add(textField11);

graceperiod = new JLabel("Grace Period");
graceperiod.setBounds(15, 375, 100, 30);
frame3.add(graceperiod);
textField14 = new JTextField();
textField14.setBounds(140, 375, 150, 30);
frame3.add(textField14);



addcredit = new JButton("Add credit card");
addcredit.setBounds( 30, 415, 150, 30);
frame3.add(addcredit);
addcredit.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent oe)
   {
      addCreditcard();
   }
});


back = new JButton("Back");
back.setBounds( 190, 415, 100, 30);
frame3.add(back);
back.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        frame1.setVisible(true);
        frame3.dispose();
    }
});

creditclear = new JButton("clear");
creditclear.setBounds( 300, 415, 100, 30);
frame3.add(creditclear);
creditclear.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        textField10.setText("");
        textField11.setText("");
        textField12.setText("");
        textField13.setText("");
        textField14.setText("");
        years2.setSelectedIndex(0);
        months2.setSelectedIndex(0);
        days2.setSelectedIndex(0);
    }
});

Displayc = new JButton("Display");
Displayc.setBounds(10, 455, 150, 30);
frame3.add(Displayc);
```

```
Displayc.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        displayCredit();
    }
});

addcreditLimit = new JButton("Add Credit Limit");
addcreditLimit.setBounds( 170, 455, 150, 30);
frame3.add(addcreditLimit);
addcreditLimit.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent oe)
    {
        creditLimit();
    }
});



cancelcredit = new JButton("Cancel credit Card");
cancelcredit.setBounds( 330, 455, 150, 30);
frame3.add(cancelcredit);
cancelcredit.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent ae)
    {
        cancelCreditCard();
```

```
            }
        });










    }
    //debit
    public int getCardID()
    {
        int CardId = INVALID;
        try {
            CardId = Integer.parseInt(textField1.getText().trim());
            if (CardId <= 0)
            {
                CardId = INVALID;
            }
        } catch (Exception ae)
        {
            JOptionPane.showMessageDialog(frame1, "INVALID ERROR" + "\n" + "Please
enter cardId in Number form",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        }
        return CardId;
```

```
}

public String getClientName()
{
    String Name = clientName.getText();
    return Name;
}



public String getIssuerBank()
{
    return textField2.getText().trim();
}

public String getBankAccount()
{
    return textField4.getText().trim();
}

public double getBalanceAmount()
{
    int balanceAmount = INVALID;
    try {
        balanceAmount = Integer.parseInt(textField5.getText().trim());
        if (balanceAmount <= 0) {
            balanceAmount = INVALID;

        }
        } catch (Exception ae) {
            JOptionPane.showMessageDialog(frame1, "INVALID INPUT" + "\n" + "Please
enter  balance amounnt",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);


        }
        return balanceAmount;
    }

    public int getPinNumber()
    {
        int pinNumber = INVALID;
        try {
            pinNumber = Integer.parseInt(textField6.getText().trim());
            if (pinNumber <= 0) {
                pinNumber = INVALID;
            }
        } catch (Exception ae) {
            JOptionPane.showMessageDialog(frame1, "INVALID INPUT" + "\n" + "Please
enter valid Pin Number",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        }
        return pinNumber;
    }

    public void addDebit()
    {
        String clientName = getClientName();
        String issuerBank = getIssuerBank();
    String bankAccount = getBankAccount();
    int cardId = getCardID();
    double balanceAmount = getBalanceAmount();
    int pinNumber = getPinNumber();

    if (clientName.isEmpty() || issuerBank.isEmpty() || bankAccount.isEmpty())
    {
      if(clientName.isEmpty())
      {
        JOptionPane.showMessageDialog(frame1, "INVALID INPUT" + "\n" + "please
enter client name", "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
      }

      if(issuerBank.isEmpty())
      {
        JOptionPane.showMessageDialog(frame1, "INVALID INPUT" + "\n" + "please
enter issuer bank name", "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
      }

      if(bankAccount.isEmpty())
      {
        JOptionPane.showMessageDialog(frame1, "INVALID INPUT" + "\n" + "please
enter bankAccount ", "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
      }
      return;
    }
```

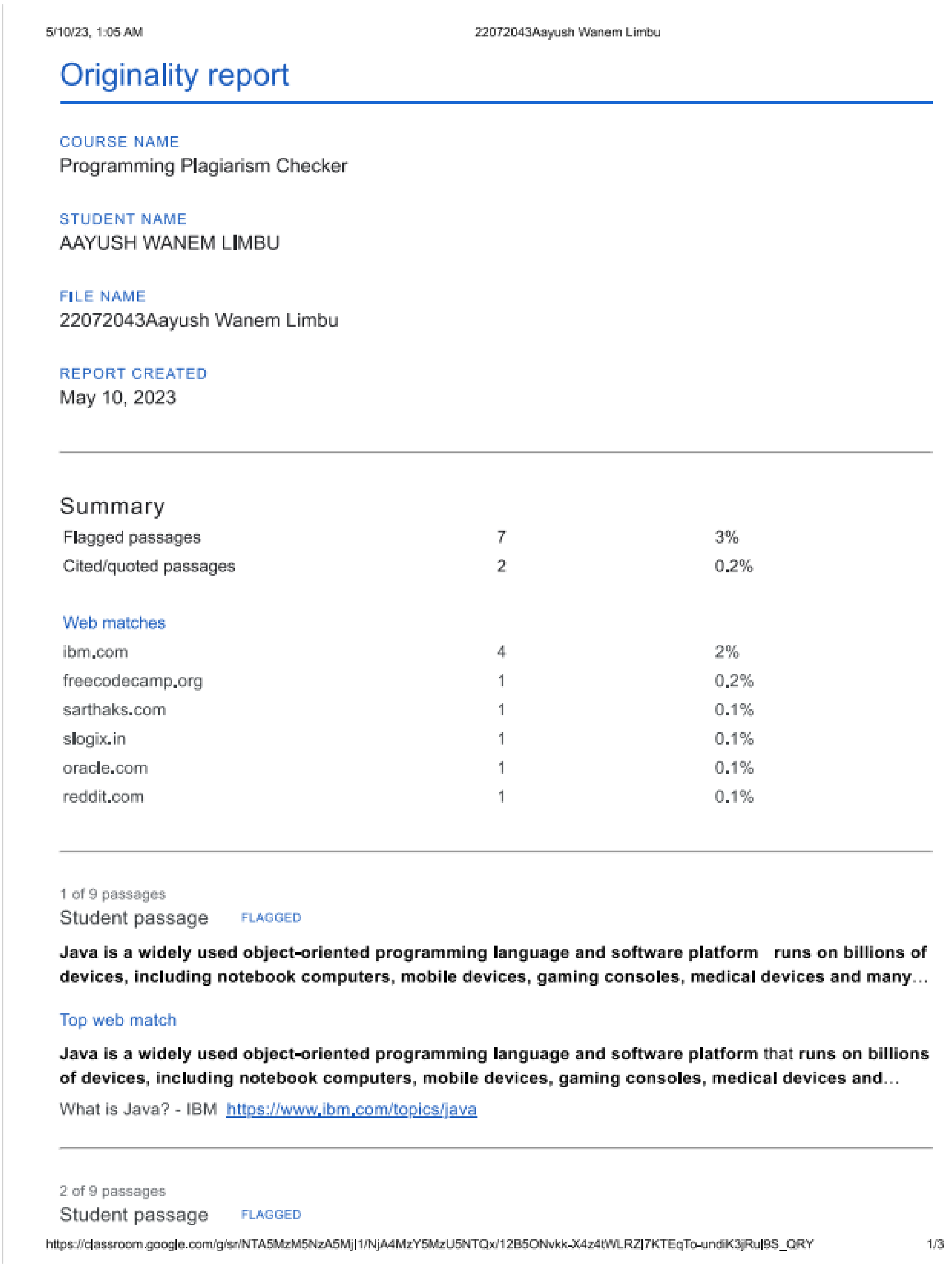
```
        if (balanceAmount < 0)
        {
            JOptionPane.showMessageDialog(frame1, "INVALID INPUT" + "\n" + "Balance
Amount cannot be less than 0", "ERROR MESSAGE",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (cardId <= 0)
        {
            JOptionPane.showMessageDialog(frame1, "INVALID INPUT" + "\n" + "CardId
cannot be less than 1",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (pinNumber <= 0 )
        {
            JOptionPane.showMessageDialog(frame1,
                    "INVALID INPUT" + "\n" + "Pin Number cannot be less than 0",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (checkCardIdUnique(cardId) == true)
        {
            array.add(new DebitCard(cardId, bankAccount, (int)balanceAmount, issuerBank,
clientName, pinNumber));
            JOptionPane.showMessageDialog(frame1, "DEBIT CARD ADDED!", "Object
Added", JOptionPane.INFORMATION_MESSAGE);
        }
        else
        {
            JOptionPane.showMessageDialog(frame1, "card id should be unique","Error
box", JOptionPane.ERROR_MESSAGE);
        }




    }
    public boolean checkCardIdUnique(int cardId)
    {
        boolean result = true;
        for (BankCard obj : array)
        {
            if (obj instanceof DebitCard)
            {
                DebitCard list = (DebitCard) obj;
                if (list.getCardId() == cardId)
                {
                    result = false;
                }
            }
        }
        return result;
    }
    public boolean checkCreditCardIdUnique(int cardId)
    {
        boolean result = true;
        for (BankCard obj : array)
        {
            if (obj instanceof CreditCard)
            {
                CreditCard list = (CreditCard) obj;
                if (list.getCardId() == cardId)
                {
                    result = false;
                }
            }
        }
        return result;
    }

    public void showDebit()
    {
        for (BankCard obj : array)
        {
            if (obj instanceof DebitCard)
            {
                DebitCard debitCard = (DebitCard) obj;
                debitCard.display();
            }
        }
    }

    //withdarw

    public int getWCardId()
```

```
{
    int cardId = INVALID;
    try {
        cardId = Integer.parseInt(textField7.getText().trim());
        if (cardId <= 0)
        {
            cardId = INVALID;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter valid Card Id",
                "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
    return cardId;
}

public int getWithdrawAmount()
{
    int withdrawamount = INVALID;
    try
    {
        withdrawamount = Integer.parseInt(textField8.getText().trim());
        if (withdrawamount <= 0)
        {
            withdrawamount = INVALID;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame2, "INVALID INPUT" + "\n" + "Please
enter valid withdrawal amount",
                "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
    return withdrawamount;
}


public int getwPinnumber()
{
    int pinNumber = INVALID;
    try {
        pinNumber = Integer.parseInt(textField8.getText().trim());
        if (pinNumber <= 0)
        {
            pinNumber = INVALID;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame2, "INVALID INPUT" + "\n" + "Please
enter valid Pin Number",
                "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
    return pinNumber;
}


public String getWithdrawalDate()
{
    String date = "";
    String year = years.getSelectedItem().toString();
    String month = months.getSelectedItem().toString();
    String day = days.getSelectedItem().toString();

    if (year.equals("year") || month.equals("month") || day.equals("day"))
    {
        date = null;
    } else
    {
        date = (year + "-" + month + "-" + day);
    }
    return date;
}

public void Withdraw()
{
    int pinNumber = getwPinnumber();
    int withdrawAmount = getWithdrawAmount();
    String withdrawalDate = getWithdrawalDate();
    int cardId = getWCardId();

    if (getWithdrawAmount() == 0)
    {
        JOptionPane.showMessageDialog(frame2, "INVALID INPUT" + "\n" + "Withdraw
Amount cannot be less than 100", "ERROR MESSAGE",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    if (getwPinnumber() <= 0 || getPinNumber()!=getwPinnumber())
    {
        JOptionPane.showMessageDialog(frame2, "INVALID INPUT" + "\n" + "enter
Correct Pin Number",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
    }


    if (getWithdrawalDate() == null)
    {
        JOptionPane.showMessageDialog(frame2, "INVALID INPUT" + "\n" + "Enter
Valid Withdrawal Date", "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (getWCardId() == 0)
    {
        JOptionPane.showMessageDialog(frame2, "INVALID INPUT" + "\n" + "Enter
Valid Card ID", "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
    }


    for (BankCard obj : array)
    {
        if (obj instanceof DebitCard)
        {
            DebitCard debitCard = (DebitCard) obj;
            if (debitCard.getCardId() == cardId)
            {
                    debitCard.withdraw(withdrawAmount, pinNumber, withdrawalDate);
                    JOptionPane.showMessageDialog(frame2, "Withdrawal Successful",
"Withdraw", JOptionPane.INFORMATION_MESSAGE);
```

```
            } else
            {
                JOptionPane.showMessageDialog(frame3, "CARD ID NOT FOUND" + "\n"
+ "Enter correct Card ID", "NOT FOUND", JOptionPane.WARNING_MESSAGE);
            }


        }
    }


}


//credit


public void showCredit()
{
    for (BankCard obj : array)
    {
        if( obj instanceof CreditCard)
        {
            CreditCard creditCard = (CreditCard) obj;
            creditCard.display();
        }
    }
}
public int getCreditCardID()
{
    int cardid = INVALID;
    try
    {
        cardid = Integer.parseInt(textField12.getText().trim());
        if (cardid <= 0)
        {
            cardid = INVALID;


        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID ERROR" + "\n" + "Please
enter valid CardId", "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
    return cardid;
}

public String getCreditClientName()
{
    return textField15.getText().trim();
}

public String getCreditIssuerBank()
{
    return textField16.getText().trim();
}

public String getCreditBankAccount()
{
    return textField17.getText().trim();


}
public double getCreditBalanceAmount()
{
    int balanceAmount = 0;
    try {
        balanceAmount = Integer.parseInt(textField18.getText().trim());
        if (balanceAmount < 0)
        {
            balanceAmount = 0;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter Balance Amount",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
    return balanceAmount;
}


public int getCVCNumber()
{
    int CVCNumber = INVALID;
    try
    {
        CVCNumber = Integer.parseInt(textField10.getText());
        if (CVCNumber < 0)
        {
            CVCNumber = INVALID;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter valid CVC Number",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }

    return CVCNumber;
}

public double getInterestRate()
{
    double interestRate = INVALID;
    try
    {
        interestRate = Double.parseDouble(textField13.getText());
        if (interestRate <= 0)
        {
            interestRate = INVALID;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter valid Interest Rate",
                    "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }

    return interestRate;
}
```

```
public String getExpirationDate()
{
    String date = "";
    String year = years2.getSelectedItem().toString();
    String month = months2.getSelectedItem().toString();
    String day = days2.getSelectedItem().toString();

    if (year.equals("year") || month.equals("month") || day.equals("day")) {
        date = null;
    }
    else
    {
        date = (year + "-" + month + "-" + day);
    }
    return date;
}


public void addCreditcard()
{
    String clientName = getCreditClientName();
    String issuerBank = getCreditIssuerBank();
    String bankAccount = getCreditBankAccount();
    int cardId = getCreditCardID();
    double balanceAmount = getCreditBalanceAmount();
    double interestRate = getInterestRate();
    int CVCNumber = getCVCNumber();
    String expirationDate = getExpirationDate();


    if (clientName.isEmpty() || issuerBank.isEmpty() || bankAccount.isEmpty())
    {
        if(clientName.isEmpty())
        {
            JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "client
Fields cannot be empty",
            "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);


            return;
        }
        if(issuerBank.isEmpty())
        {
            JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "issuer
Fields cannot be empty",
            "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);


            return;
        }
        if(bankAccount.isEmpty())
        {
            JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "bank
Fields cannot be empty",
            "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);


            return;
        }
        return;
    }


    if (CVCNumber <= 0)
    {
        JOptionPane.showMessageDialog(frame3,
        "INVALID INPUT" + "\n" + "CVC Number cannot be less than 1", "ERROR
MESSAGE",
        JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (balanceAmount < 0)
    {
        JOptionPane.showMessageDialog(frame3,
        "INVALID INPUT" + "\n" + "Balance Amount cannot be less than 0", "ERROR
MESSAGE",
        JOptionPane.ERROR_MESSAGE);
        return;
    }


    if (expirationDate == null)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Enter
Valid Expiration Date",
        "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
    }


    if (cardId <= 0)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "CardId
cannot be less than 1",
        "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if (interestRate <= 0)
     {
        JOptionPane.showMessageDialog(frame3,
        "INVALID INPUT" + "\n" + "Rate cannot be less than 1",
        "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if (checkCreditCardIdUnique(cardId) == true )
    {
        array.add(new CreditCard(cardId,  issuerBank, bankAccount, (int)
balanceAmount,  CVCNumber,  interestRate,  expirationDate, clientName));
        JOptionPane.showMessageDialog(frame3,
        "CREDITCARD ADDED" + "\n" + "Client Name: " + clientName + "\n" + "Card ID:
" + cardId +"\n"+ "Issuer Bank: "
        + issuerBank + "\n" + "Bank Account: " + bankAccount + "\n" + "BalanceAmount:
" + balanceAmount + "\n"
        + "CVC Number: " + CVCNumber + "\n" + "Interest Rate: " + interestRate + "\n"
+ "Expiration Date: "
        + expirationDate,
        "DETAILS OF CARD", JOptionPane.INFORMATION_MESSAGE);
    }

    else
    {
        JOptionPane.showMessageDialog(frame3, "card id should be unique",
        "Error box", JOptionPane.ERROR_MESSAGE);
    }
}
public void displayCredit()
{
    for (BankCard obj : array)
    {
        if( obj instanceof CreditCard)
        {
            CreditCard creditCard = (CreditCard) obj;
            creditCard.display();
        }



    }
}

//credit limit


public int getCardid()
{
    int cardid = INVALID;
    try
    {
        cardid = Integer.parseInt(textField12.getText());
        if (cardid <= 0)
        {
            JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "CardId
cannot be less than 1",
                    "ERROR MESSAGE", JOptionPane.INFORMATION_MESSAGE);


        }
    } catch (Exception ae) {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter valid Card ID",
                "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
    return cardid;
}


public double getCreditLimit()
{
    double creditLimit = INVALID;
    try
    {
        creditLimit = Double.parseDouble(textField18.getText());
        if (creditLimit <= 0)
        {
            creditLimit = INVALID;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter valid CreditLimit",
                "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }

    return creditLimit;
}

public int getGracePeriod()
{
    int graceperiod = INVALID;
    try
    {
        graceperiod = Integer.parseInt(textField14.getText());
        if (graceperiod <= 0)
        {
            graceperiod = INVALID;
        }
    } catch (Exception ae)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter valid GracePeriod",
                "ERROR MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
    return graceperiod;
}

public void creditLimit()
{
    int cardid = getCardid();
    double creditlimit = getCreditLimit();
    int gracePeriod = getGracePeriod();
    if (cardid == INVALID)
    {
        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "CardId
cannot be less than 1",
                "ERROR MESSAGE", JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    if (creditlimit == INVALID)
    {

        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Credit
Limit cannot be less than 1",
                "ERROR MESSAGE", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    if (gracePeriod == INVALID)
    {

        JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Grace
Period cannot be less than 1",
                "ERROR MESSAGE", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    for (BankCard obj : array)
    {
        if (obj instanceof CreditCard)
        {
            CreditCard creditCard = (CreditCard) obj;
            if (creditCard.getCardId() == cardid)
            {
                creditCard.setCreditLimit(creditlimit, gracePeriod);
                JOptionPane.showMessageDialog(frame3, "Credit Limit has been set",
                "Credit Limit", JOptionPane.INFORMATION_MESSAGE);
            } else
            {
                JOptionPane.showMessageDialog(frame3, "CARD ID NOT FOUND" + "\n"
+ "Enter correct Card ID",
                        "NOT FOUND", JOptionPane.WARNING_MESSAGE);
            }


        }
    }
}



//#region for cancel credit card


    public int getCancelCardId()
    {
        int cardId = INVALID;
        try
        {
            cardId = Integer.parseInt(textField12.getText());
            if (cardId <= 0)
            {
                cardId = INVALID;
            }
        } catch (Exception e)
        {
            JOptionPane.showMessageDialog(frame3, "INVALID INPUT" + "\n" + "Please
enter valid card Id",
                    "ERROR MESSAGE", JOptionPane.INFORMATION_MESSAGE);
        }
        return cardId;
    }

    public void cancelCreditCard()
    {
        boolean isfound = false;
        int cardID = getCancelCardId();

        for(BankCard obj: array)
        {
            if(obj instanceof CreditCard)
            {
                CreditCard creditCard = (CreditCard) obj;
                if(creditCard.getCardId()==cardID)
                {
                    isfound = true;
                    creditCard.cancelCreditCard();
                    break;
                }
            }
        }
        if (isfound == true)
        {
            JOptionPane.showMessageDialog(frame3, "Canceled credit card", "success
box", JOptionPane.INFORMATION_MESSAGE);
        }
        else
        {
            JOptionPane.showMessageDialog(frame2, "cardId doesnt match", "error Box",
                    JOptionPane.INFORMATION_MESSAGE);
        }
    }


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

10. Originality Report



5/10/23, 1:05 AM 22072043Aayush Wanem Limbu

Originality report

COURSE NAME
Programming Plagiarism Checker

STUDENT NAME
AAYUSH WANEM LIMBU

FILE NAME
22072043Aayush Wanem Limbu

REPORT CREATED
May 10, 2023

Summary

Flagged passages	7	3%
Cited/quoted passages	2	0.2%
Web matches		
ibm.com	4	2%
freecodecamp.org	1	0.2%
sarthaks.com	1	0.1%
slogix.in	1	0.1%
oracle.com	1	0.1%
reddit.com	1	0.1%

1 of 9 passages
Student passage FLAGGED

Java is a widely used object-oriented programming language and software platform runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many...

Top web match

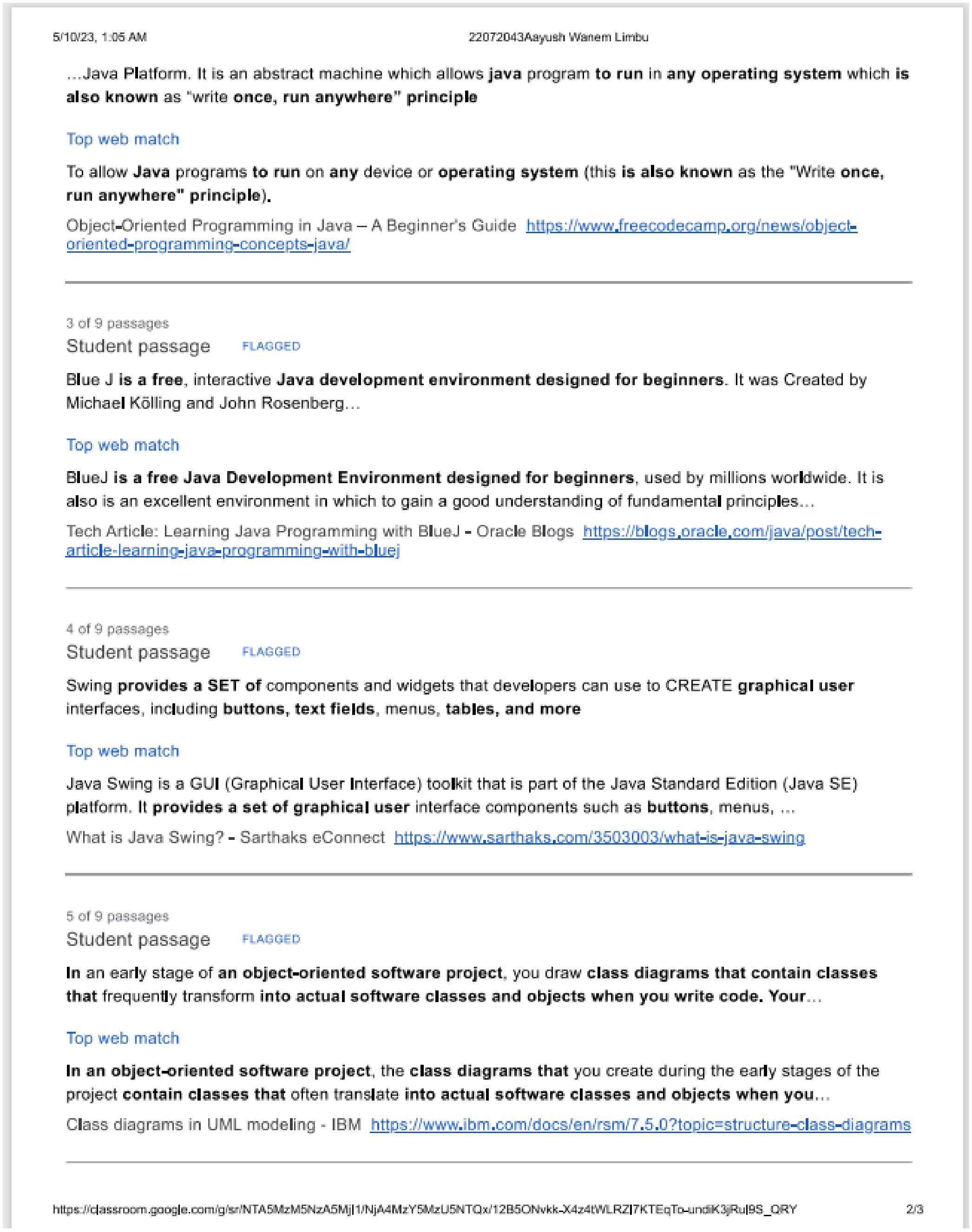
Java is a widely used object-oriented programming language and software platform that **runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and...**

What is Java? - IBM https://www.ibm.com/topics/java

2 of 9 passages
Student passage FLAGGED

https://classroom.google.com/g/sr/NTA5MzM5NzA5MjI1/NjA4MzY5MzU5NTQx/12B5ONvkk-X4z4tWLRZI7KTEqTo-undiK3jRuI9S_QRY 1/3

Figure 40 Originality test 1



5/10/23, 1:05 AM 22072043Aayush Wanem Limbu

…Java Platform. It is an abstract machine which allows **java** program **to run** in **any operating system** which **is also known** as "write **once, run anywhere" principle**

Top web match

To allow **Java** programs **to run** on **any** device or **operating system** (this **is also known** as the "Write **once, run anywhere" principle**).

Object-Oriented Programming in Java – A Beginner's Guide https://www.freecodecamp.org/news/object-oriented-programming-concepts-java/

3 of 9 passages

Student passage FLAGGED

Blue J **is a free**, interactive **Java development environment designed for beginners**. It was Created by Michael Kölling and John Rosenberg…

Top web match

BlueJ **is a free Java Development Environment designed for beginners**, used by millions worldwide. It is also is an excellent environment in which to gain a good understanding of fundamental principles…

Tech Article: Learning Java Programming with BlueJ - Oracle Blogs https://blogs.oracle.com/java/post/tech-article-learning-java-programming-with-bluej

4 of 9 passages

Student passage FLAGGED

Swing **provides a SET of** components and widgets that developers can use to CREATE **graphical user** interfaces, including **buttons, text fields**, menus, **tables, and more**

Top web match

Java Swing is a GUI (Graphical User Interface) toolkit that is part of the Java Standard Edition (Java SE) platform. It **provides a set of graphical user** interface components such as **buttons**, menus, …

What is Java Swing? - Sarthaks eConnect https://www.sarthaks.com/3503003/what-is-java-swing

5 of 9 passages

Student passage FLAGGED

In an early stage of **an object-oriented software project**, you draw **class diagrams that contain classes that** frequently transform **into actual software classes and objects when you write code. Your**…

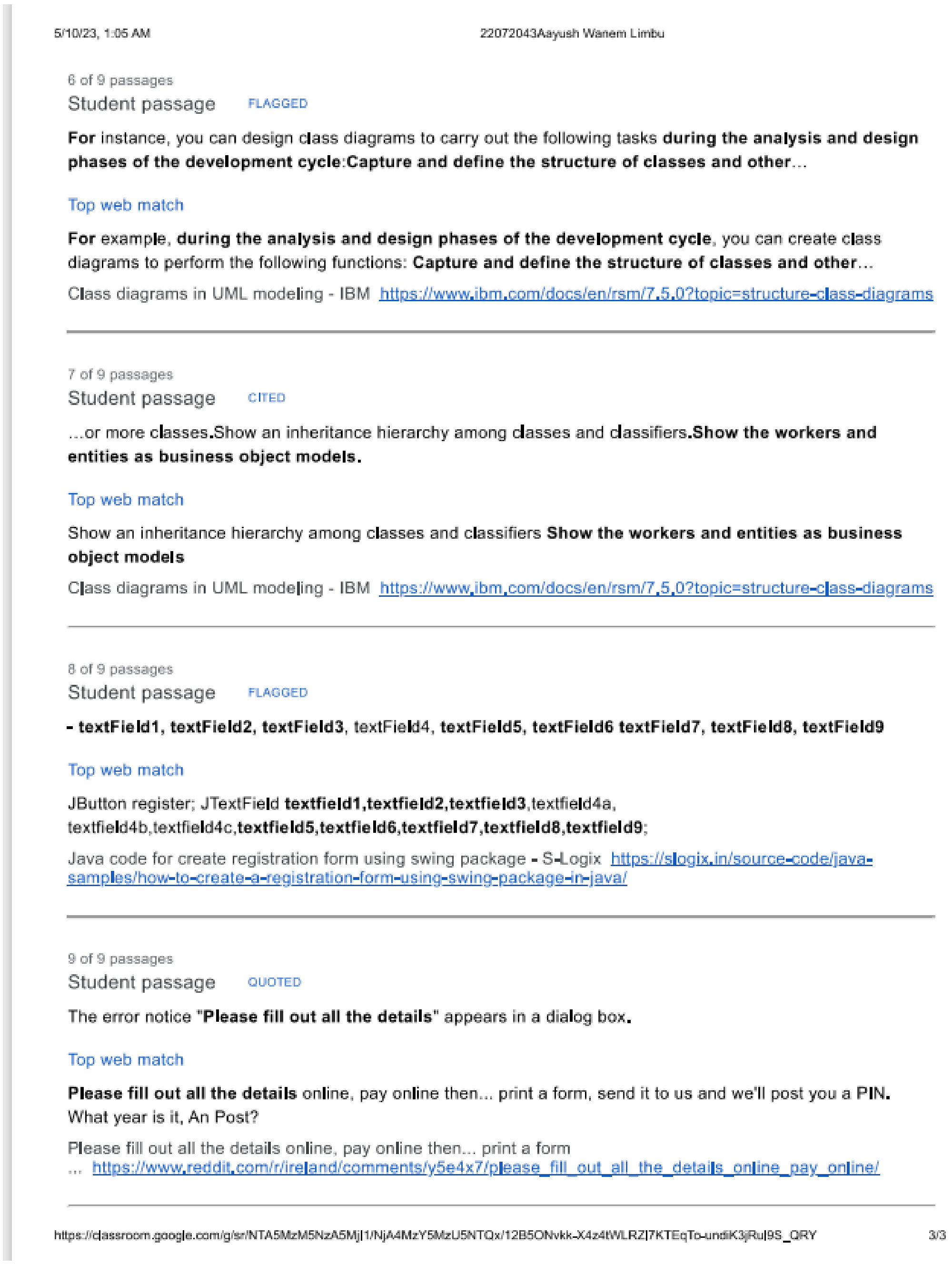
Top web match

In an object-oriented software project, the **class diagrams that** you create during the early stages of the project **contain classes that** often translate **into actual software classes and objects when you**…

Class diagrams in UML modeling - IBM https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams

https://classroom.google.com/g/sr/NTA5MzM5NzA5Mjl1/NjA4MzY5MzU5NTQx/12B5ONvkk-X4z4tWLRZI7KTEqTo-undiK3jRuI9S_QRY 2/3

Figure 41 Originality test 2



5/10/23, 1:05 AM 22072043Aayush Wanem Limbu

6 of 9 passages

Student passage FLAGGED

For instance, you can design class diagrams to carry out the following tasks **during the analysis and design phases of the development cycle**:**Capture and define the structure of classes and other**…

Top web match

For example, **during the analysis and design phases of the development cycle**, you can create class diagrams to perform the following functions: **Capture and define the structure of classes and other**…

Class diagrams in UML modeling - IBM https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams

7 of 9 passages

Student passage CITED

…or more classes.Show an inheritance hierarchy among classes and classifiers.**Show the workers and entities as business object models.**

Top web match

Show an inheritance hierarchy among classes and classifiers **Show the workers and entities as business object models**

Class diagrams in UML modeling - IBM https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams

8 of 9 passages

Student passage FLAGGED

- **textField1, textField2, textField3**, textField4, **textField5, textField6 textField7, textField8, textField9**

Top web match

JButton register; JTextField **textfield1,textfield2,textfield3**,textfield4a, textfield4b,textfield4c,**textfield5,textfield6,textfield7,textfield8,textfield9**;

Java code for create registration form using swing package - S-Logix https://slogix.in/source-code/java-samples/how-to-create-a-registration-form-using-swing-package-in-java/

9 of 9 passages

Student passage QUOTED

The error notice "**Please fill out all the details**" appears in a dialog box.

Top web match

Please fill out all the details online, pay online then... print a form, send it to us and we'll post you a PIN. What year is it, An Post?

Please fill out all the details online, pay online then... print a form … https://www.reddit.com/r/ireland/comments/y5e4x7/please_fill_out_all_the_details_online_pay_online/

https://classroom.google.com/g/sr/NTA5MzM5NzA5MjI1/NjA4MzY5MzU5NTQx/12B5ONvkk-X4z4tWLRZI7KTEqTo-undiK3jRuI9S_QRY 3/3

Figure 42 Originality test 3