



 slington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Dikshya Sharma

London Met ID: 22067520

College ID: NP01CP4A220029

Assignment Due Date: Friday, January 27, 2023

Assignment Submission Date: Wednesday, January 25, 2023

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

Table of Contents

TABLE OF FIGURES.....	4
Table of Table.....	5
1. INTRODUCTION	1
2. TOOLS USED FOR COURSEWORK	2
a. BlueJ.....	2
b. MS-Word.....	2
c. Draw.io	3
3. CLASS DIAGRAM:	4
4. PSEUDOCODE	6
4.1 For Bank Card Class.....	6
4.2 For Debit Card Class	10
4.3 For Credit Card Class	14
5. METHOD DESCRIPTION	19
5.1 For Bank Card Class.....	19
5.2 For Debit Card Class	21
5.3 For Credit Card Class	23
6. TESTING	25
6.1 Test 1.....	25
6.2 Test 2.....	28
6.3 Test 3.....	31
7. ERROR DETECTION	37
7.1 Error 1:.....	37
Correction Of Error 1:.....	37
7.2 Error 2:.....	39
Correction Of Error 2:.....	39
7.3 Error 3:.....	41
Correction Of Error 3:.....	42
8. CONCLUSION.....	43
References	44
APPENDIX.....	45
a. For BankCard Class.....	45

b.	For DebitCard Class	49
c.	For CreditCard Class.....	53

TABLE OF FIGURES

Figure 1: Class Diagram of Bank Card Class	4
Figure 2: Class Diagram of Debit Card Class	4
Figure 3: Class Diagram of Credit Card Class	4
Figure 4: Inheritance Diagram of Class Credit Card and Debit Card	5
Figure 5: Inserting data in the attributes of DebitCard Class	26
Figure 6: Inspecting Debit Card Class	26
Figure 7: Withdrawing the amount	27
Figure 8: Inspecting the DebitCard Class after amount withdrawal	27
Figure 9: Inserting data in the attributes of CreditCard Class	29
Figure 10: Inspecting DebitCard Class	29
Figure 11: Setting the Credit Limit	30
Figure 12: Inspecting CreditCard Class after setting credit limit	30
Figure 13: Inserting data in the attributes of CreditCard Class (Test 3)	32
Figure 14: Inspecting CreditCard Class (Test 3)	32
Figure 15: Cancelling Credit Card	33
Figure 16: Inspecting CreditCard Class after cancelling credit card	33
Figure 17: Inserting data in the attributes of DebitCard Class (Test 4)	35
Figure 18: Displaying the DebitCard Class	35
Figure 19: Inserting data in the attributes of CreditCard Class (Test 4)	36
Figure 20: Displaying the DebitCard Class	36
Figure 21: Syntax Error	37
Figure 22: Correction of Syntax Error	38
Figure 23: Semantic Error	39
Figure 24: Correction of Semantic Error	40
Figure 25: Logical Error	41
Figure 26: Correction of Logical Error	42

Table of Table

Table 1: Inspect the debit card class, withdraw the amount and re-inspect the debit card class	25
Table 2:Inspect the credit card class, set the credit limit and re-inspect the credit card class	28
Table 3:Inspect the credit card class again after cancelling the credit card	31
Table 4:Display the details of debit card and credit card	34

1. INTRODUCTION

This coursework aims on creating a system about Bank Cards which is categorized as Credit Cards and Debit Cards. The main objective of this report is to document information about the project's goal which includes implementation, testing and the issues encountered. This project demonstrates the concepts of Object-Oriented Programming and encapsulation of data. This coursework uses BlueJ as it's IDE. This project includes three different classes namely:

1. Bank Card Class
2. Debit Card Class
3. Credit Card Class

The Bank Card class is the super class which has Credit Card and Debit Card sub classes. Every class has several methods which includes constructors, accessors mutators methods and methods to display data within each class. This system processes transactions, manages accounts, checks credit balance, withdraw funds and so on.

2. TOOLS USED FOR COURSEWORK

Various tools were used while doing this coursework which includes BlueJ, Ms-Word and Draw.io. These tools are shortly described below:

a. BlueJ



BlueJ is an integrated Java development environment developed by the University of Kent, in the UK which is built on top of a standard Java SDK and thus uses a standard compiler and virtual machine. It is an excellent environment to gain a good understanding of fundamental principles of Object-Oriented (Kolling, et al., 2010).

b. MS-Word



Microsoft Word is a word processor published by Microsoft. It is one of the office productivity applications included in the Microsoft Office suite. Charles Simonyi and Richard Brodie developed the initial version of Microsoft Word, which was first launched in 1983 (Anon., n.d.).

c. Draw.io

draw.io is a proprietary software for making diagrams and charts. The software gives user the option to design their own layout or use the automatic layout feature. It contains a large selection of shapes and hundreds of visual elements that makes diagram or chart one of a kind (Anon., n.d.).

3. CLASS DIAGRAM:

Class diagram for this project is given below:

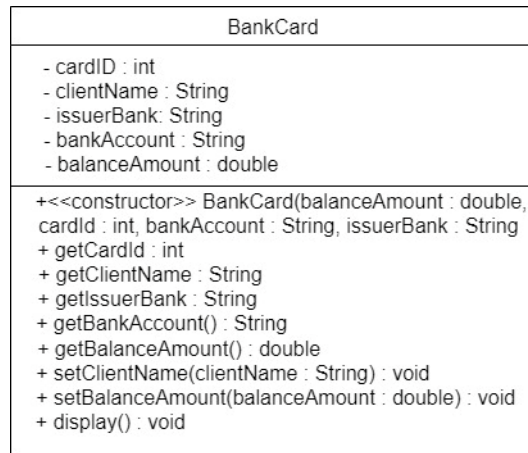


Figure 1: Class Diagram of Bank Card Class

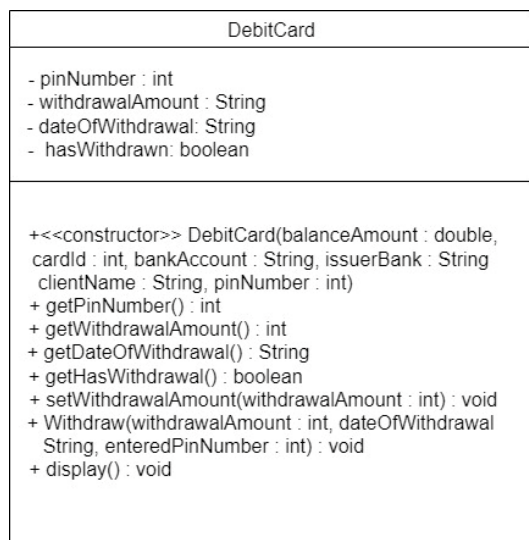


Figure 2: Class Diagram of Debit Card Class

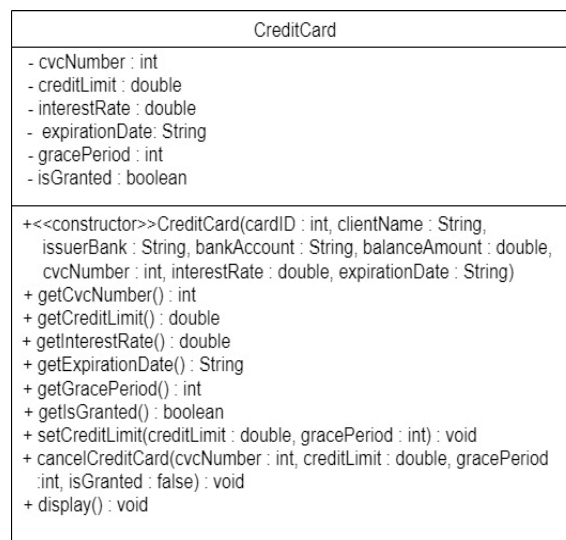


Figure 3: Class Diagram of Credit Card Class

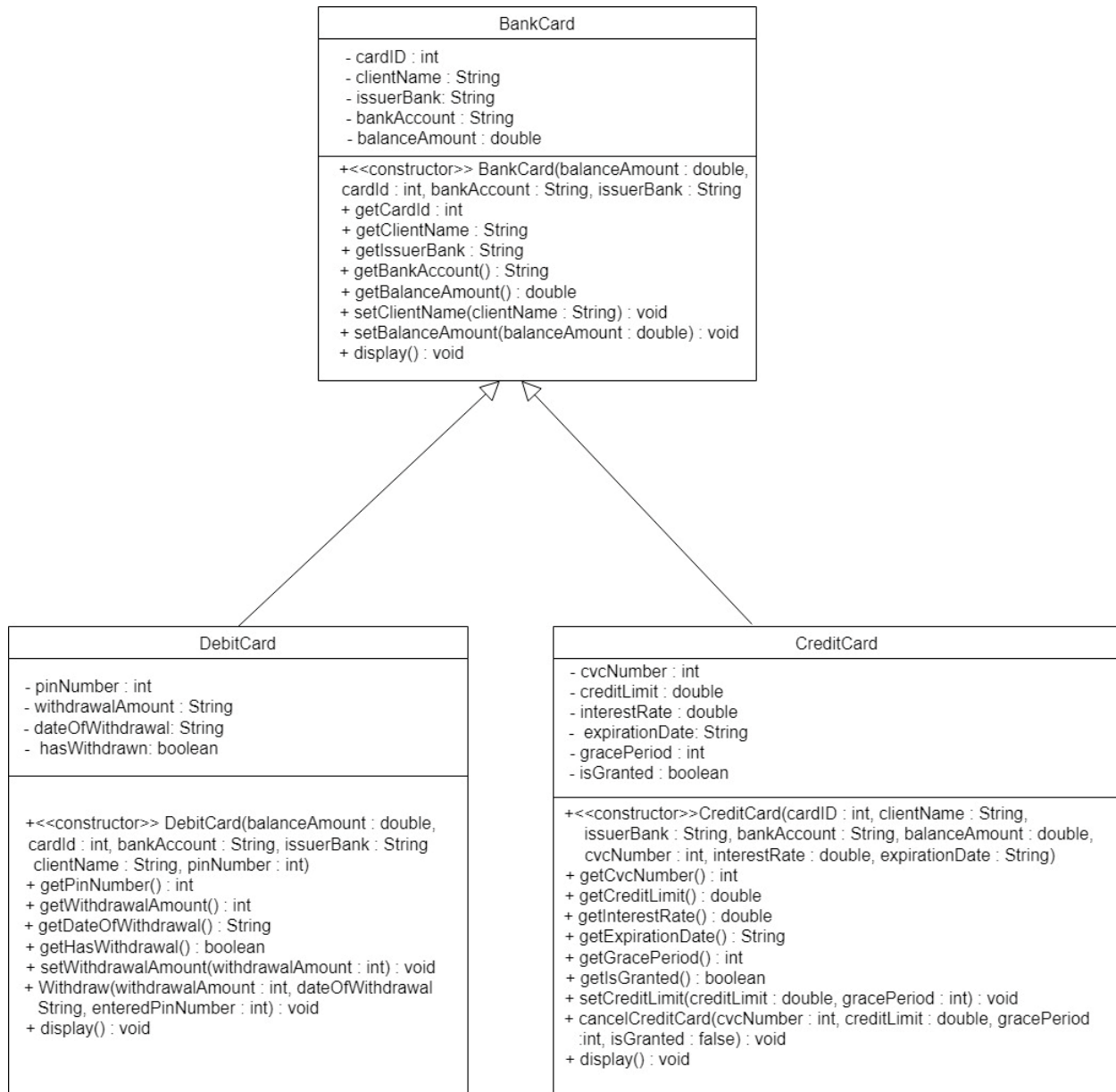


Figure 4: Inheritance Diagram of Class Credit Card and Debit Card

4. PSEUDOCODE

PseudoCode for this project is given below:

4.1 For Bank Card Class

CREATE a parent class BankCard

DO

DECLARE an attribute cardId of type integer

DECLARE an attribute clientName of type string

DECLARE an attribute issuerBank of type string

DECLARE an attribute bankAccount of type string

DECLARE an attribute balanceAmount of type double

CREATE a constructor BankCard that accepts four parameters (balanceAmount of type double, cardID of type integer, bankAccount of type String, issuerBank of type string)

DO

ASSIGN parameter of balanceAmount to attribute of balanceAmount

ASSIGN parameter of cardId to attribute of cardID

ASSIGN parameter of bankAccount to attribute of bankAccount

ASSIGN parameter of issuerBank to attribute of issuerBank

ASSIGN parameter of clientName to empty string

END DO

CREATE an accessor method getCardId() with return type integer

DO

RETURN cardId

END DO

CREATE an accessor method getClientName() with return type string

DO

RETURN clientName

END DO

CREATE an accessor method getBankAccount () with return type string

DO

RETURN bankAccount

END DO

CREATE an accessor method getBalanceAmount () with return type double

DO

RETURN balanceAmount

END DO

CREATE a mutator method setClientName with the parameter (clientName of type string) with return type void

DO

RETURN clientName

END DO

CREATE a mutator method setBalanceAmount with the parameter (balanceAmount of type double) with return type void

DO

RETURN balanceAmount

END DO

CREATE a display () method with return type void

DO

DISPLAY ("Card ID:" + value stored in an attribute cardId)

IF (clientName is empty ())

THEN

DISPLAY ("Client name is not assigned")

ELSE

DISPLAY ("Client Name:" + value stored in an attribute clientName)

END IF

DISPLAY ("Issuer Bank:" + value stored in an attribute issuerBank)

DISPLAY (Bank Account:" + value stored in an attribute
bankAccount)

DISPLAY ("Balance Amount:" +value stored in an attribute
balanceAmount)

END DO

END DO

4.2 For Debit Card Class

CREATE a child class DebitCard of parent class BankCard

DO

DECLARE an attribute pinNumber of type integer

DECLARE an attribute withdrawalAmount of type integer

DECLARE an attribute dateOfWithdrawal of type String

DECLARE an attribute hasWithdrawn of type Boolean

CREATE a constructor that accepts six parameters (balanceAmount of type double, cardId of type integer, bankAccount of type string, issuerBank of type string, clientName of type String, pinNumber of type integer)

DO

CALL a superclass constructor with four parameters with same order (balanceAmount, cardId, bankAccount, issuerBank)

CALL a superclass mutator method setClientName with parameter clientName

ASSIGN parameter of pinNumber to attribute of pinNumber

ASSIGN parameter of hasWithdrawn to attribute of hasWithdrawn and set value (flag) of attribute hasWithdrawn as false

END DO

CREATE an accessor method getPinNumber () with return type integer

DO

RETURN pinNumber

END DO

CREATE an accessor method getWithdrawalAmount () with return type integer

DO

```
        RETURN withdrawalAmount
    END DO
```

```
    CREATE an accessor method getDateOfWithdrawal () with return type
    string
    DO
        RETURN dateOfWithdrawal
    END DO
```

```
    CREATE an accessor method getHasWithdrawn () with return type Boolean
    DO
        RETURN hasWithdrawn
    END DO
```

```
    CREATE a mutator method setWithdrawalAmount with the parameter
    (withdrawalAmount of type integer) with return type void
    DO
        RETURN withdrawalAmount
    END DO
```

```
    CREATE a method name Withdraw with three parameters
    (withdrawalAmount of type integer, dateOfWithdrawal of type string,
    pinNumber of type integer) with return type void
    DO
        IF (pinNumber is equal with value of set parameter of pinNumber )
        THEN
            IF(withdrawalAmount is less than or equal to accessor method of
            getBalanceAmount from a super class)
            THEN
```



```
    CALL A mutator method of setBalanceAmount of super class with  
    parameter (withdrawalAmount substarcted from value stored on  
    accessor method getBalanceAmount())  
    SET parameter withdrawalAmount to the attribute of  
    withdrawalAmount  
    SET parameter dateOfWithdrawal to the attribute of  
    dateOfWithdrawal  
    SET parameter hasWithdrawn to attribute hasWithdrawn and set the  
    value (flag) of attribute hasWithdrawn as true  
    DISPLAY("Successful transaction, remaining amount : " +  
    super.getBalanceAmount())  
ELSE  
    DISPLAY ("Insufficient Balance!")  
END IF  
END DO  
  
ELSE  
    DISPLAY ("IN Valid Pin Number!")  
END IF  
END DO
```

```
CREATE a display () method with return type void  
DO  
    DISPLAY the display () method of superclass BankCard  
    DISPLAY ("PIN-Number:" + value stored in an attribute pinNumber)  
    IF (hasWithdrawn is set to true value)  
    THEN  
        DISPLAY ("Withdrawal-Amount:" + value of an attribute withdrawalAmount)  
        DISPLAY ("Date-Of-Withdrawal:" + value of an attribute dateOfWithdrawal)  
    ELSE
```

DISPLAY ("No withdrawal have taken place as of yet! Balance Amount:" +
value of an attribute balanceAmount from super class)

END IF

END DO

4.3 For Credit Card Class

CREATE a child class CreditCard of parent class

DO

DECLARE an attribute cvcNumber of type integer

DECLARE an attribute creditLimit of type double

DECLARE an attribute interestRate of type double

DECLARE an attribute expirationDate of type string

DECLARE an attribute gracePeriod of type Integer

DECLARE an attribute isGranted of type Boolean

CREATE a constructor that accepts eight parameter (cardId of type integer, clientName of type string, issuerBank of type string, bankAccount of type string, balanceAmount of type double, cvcNumber of type integer, interestRate of type double, expirationDate of type string)

DO

CALL a superclass constructor with four parameter with same order

(balanceAmount, cardId, bankAccount, issuerBank)

CALL a setter method setClientName to set the value of clientName i.e. setClientName (clientName) from superclass

ASSIGN parameter of cvcNumber to attribute of cvcNumber

ASSIGN parameter of interestRate to attribute of interestRate

ASSIGN parameter of expirationDate to attribute of expirationDate

ASSIGN parameter of isGranted to set flag to false

END DO

CREATE an accessor method getCvcNumber () with return type integer

DO

RETURN cvcNumber

END DO

CREATE an accessor method getCreditLimit () with return type double

DO

RETURN creditLimit

END DO

CREATE an accessor method getInterestRate () with return type double

DO

RETURN interestRate

END DO

CREATE an accessor method getExpirationDate () with return type string

DO

RETURN expirationDate

END DO

CREATE an accessor method getGracePeriod () with return type integer

DO

RETURN gracePeriod

END DO

CREATE an accessor method getIsGranted () with return type Boolean

DO

RETURN isGranted

END DO

CREATE a mutator method setCreditLimit with the parameter (creditLimit of type double and gracePeriod of type integer) with return type void

DO

ASSIGN parameter of creditLimit to attribute of creditLimit

ASSIGN parameter of gracePeriod to attribute of gracePeriod

IF (creditLimit is less than or equal to 2.5 times the balanceAmount i.e.(2.5* getBalanceAmount()))

THEN

DISPLAY("Credit is successfully granted!")

SET parameter isGranted with value true

ELSE

DISPLAY("Credit couldn't be issued due to a credit limit overused")

END IF

END DO

CREATE a CancelCreditCard method() with return type void

DO

SET parameter cvcNumber with value zero

SET parameter creditLimit with value zero

SET parameter gracePeriod with value zero

SET parameter isGranted with value false

END DO

CREATE a display () method with return type void

DO

DISPLAY the display () method of superclass BankCard

IF (isGranted is set to true value)

THEN

DISPLAY ("Credit limit is valid hence credit is granted")

DISPLAY ("Credit Limit:" + value of attribute creditLimit)

DISPLAY ("Grace Period:" + value of attribute gracePeriod)

ELSE

DISPLAY("Credit is not granted.")

END IF

DISPLAY("CVC number:" + value of attribute cvcNumber)

DISPLAY ("Interest Rate:" + value of attribute interestRate)

DISPLAY ("Expiration Date:" + value of attribute expirationDate)

END DO

END DO

5. METHOD DESCRIPTION

The methods of each class are described below:

5.1 For Bank Card Class

BankCard(double balanceAmount, int cardId, string bankAccount, string issuerBank)

This method is used to initialize the object's state. It initializes the properties of a bank card object such as balanceAmount, cardId, bankAccount and issuerBank. It also sets the default value of clientName to empty.

getCardId()

This method is used get the value of Id of Card.

getClientName()

This method is used to get the value of name of Client.

getIssuerBank()

This method is used to get the value of Issuer Bank.

getBankAccount()

This method is used to get the value of Bank Account.

getBalanceAmount()

This method is used to get the value of Balance Amount.

setClientName(String clientName)

This method is used to set the value of Client Name.

setBalanceAmount(double balanceAmount)

This method is used to set the value of balance amount.

display()

This method displays the cardId, issuerBank, bankAccount, balanceAmount and if the clientName is empty it displays client name not assigned message else it displays the clientName with all suitable annotation.

5.2 For Debit Card Class

DebitCard(double balanceAmount, int cardId, string bankAccount, string issuerBank, string clientName, int pinNumber)

This method is used to initialize the object's state. It initializes the object pinNumber of credit card and calls the method of super class Bank Card and initializes the object balanceAmount, cardId, bankAccount, and issuerBank. It also sets the hasWithdrawan flag to false.

getPinNumber()

This method is used to get the value of Pin Number.

getWithdrawalAmount()

This method is used to get the value of Withdrawal Amount.

getDateOfWithdrawal()

This method is used to get the value of Date of Withdrawal.

getHasWithdrawan()

This method is used to get the value of Withdrawal status.

setWithdrawalAmount(int withdrawalAmount)

This method is used to set the value of withdrawalAmount.

Withdraw(int withdrawalAmount, string dateOfWithdrawal, int pinNumber)

This method is used to make a withdraw. If pin number is correct and withdrawalAmount is less than or equal to balanceAmount then it allows the transaction and displays suitable message with remaining Balance Amount else it displays invalid pin number or insufficient balance message.

display()

This method displays pinNumber and if any withdrawal is performed, it displays withdrawalAmount with dateOfWithdrawal else it shows the no any withdrawal message with existing balanceAmount.

5.3 For Credit Card Class

CreditCard(int cardID, String clientName, String issuerBank, String bankAccount, double balanceAmount, int cvcNumber, double interestRate, String expirationDate)

This method is used to initialize the object's state. It calls the method from super class and initializes cvcNumber, interestRate, expirationDate, and sets isGranted flag to false.

getCvcNumber()

This method is used to get the value of CVC Number.

getCreditLimit()

This method is used to get the value of Credit Limit.

getInterestRate()

This method is used to get the value of Interest Rate.

getExpirationDate()

This method is used to get the value of Expiration Date.

getGracePeriod()

This method is used to get the value of Grace Period.

getIsGranted()

This method is used to get the status of Granted.

setCreditLimit(double creditLimit, int gracePeriod)

This method is used to set the value of CreditLimit. If credit limit is less than or equals to 2.5 times the balanceAmount then it displays the credit successfully granted message and else it displays Credit couldn't be issued message.

`cancelCreditCard()`

It is used to cancel the credit Card.

`display()`

This method displays cvc number, interest rate and expiration date. If `isGranted` flag is true it also displays credit limit and grace period with credit limit valid message else displays credit is not granted message.

6. TESTING

Following are the testing for the given coursework.

6.1 Test 1

Test No.	1
Objective:	To inspect the debit card class, withdraw the amount and re-inspect the debit card class
Action:	<p>➔ The debit card is called with the following arguments:</p> <p>balanceAmount : 50000</p> <p>cardId : 5</p> <p>bankAccount : "09758456234"</p> <p>issuerBank : "Everest Bank"</p> <p>clientName : "Smith"</p> <p>pinNumber : 7179</p> <p>➔ Inspection of the debit card class.</p> <p>➔ void withdrawalAmount is called with the following arguments:</p> <p>withdrawalAmount : 7000</p> <p>dateOfWithdrawal : "2022-04-05"</p> <p>pinNumber : 7179</p> <p>➔ Re- inspection of the debit card class.</p>
Expected Result :	The amount would be withdraw.
Actual Result :	The amount was withdrawan.
Conclusion:	The test is successful.

Table 1: Inspect the debit card class, withdraw the amount and re-inspect the debit card class

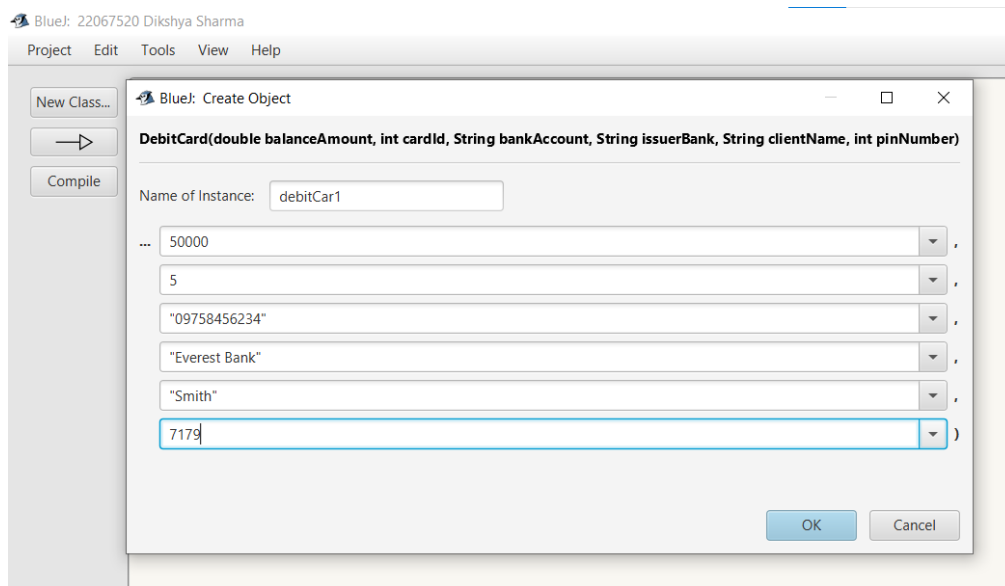


Figure 5: Inserting data in the attributes of DebitCard Class

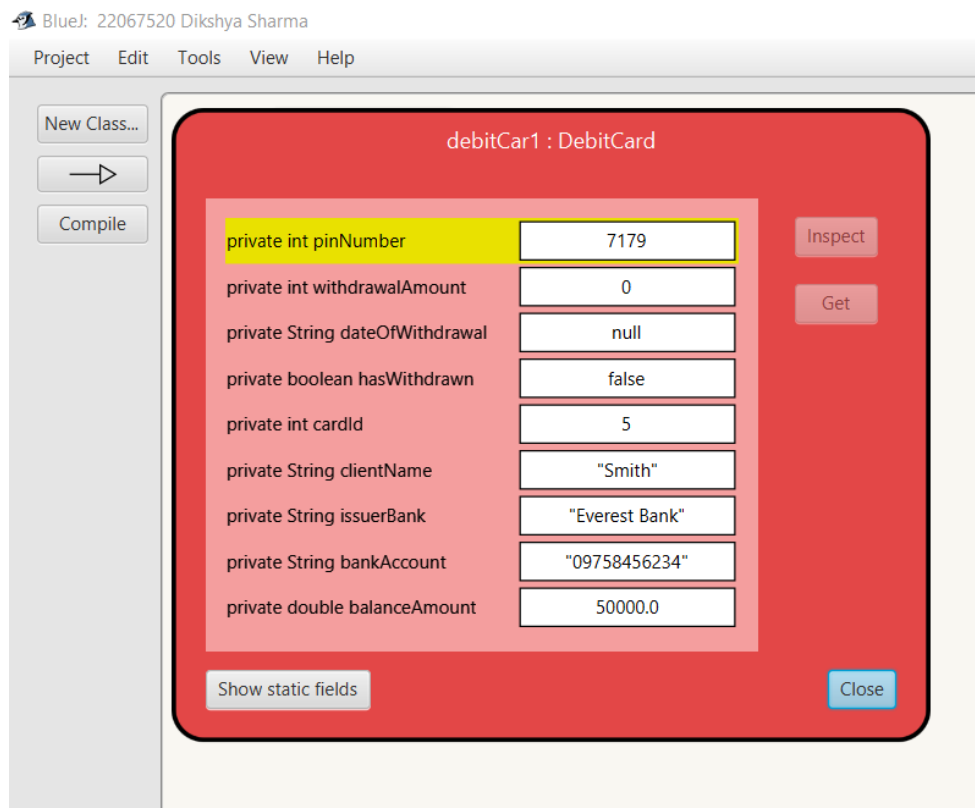


Figure 6: Inspecting Debit Card Class

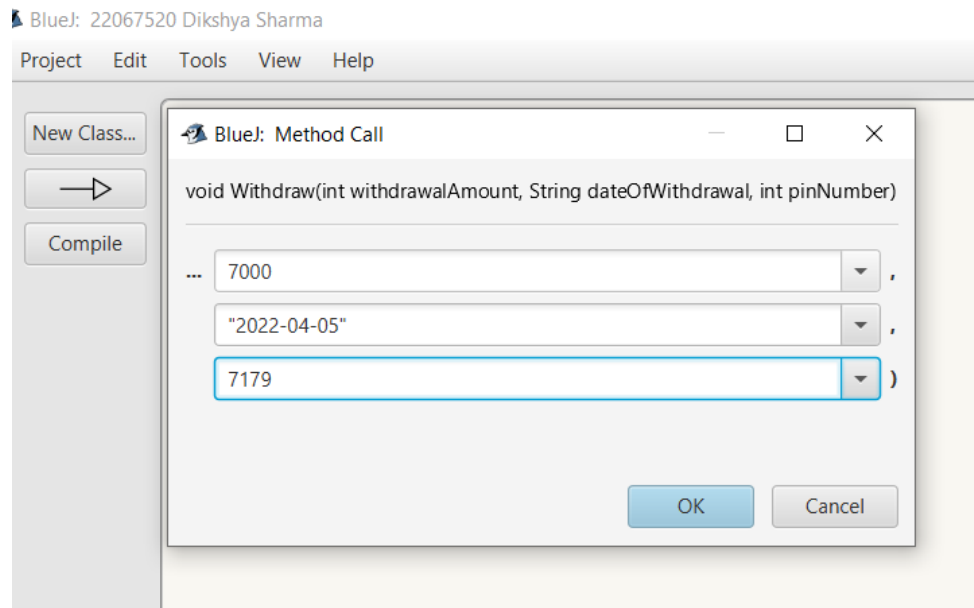


Figure 7: Withdrawing the amount

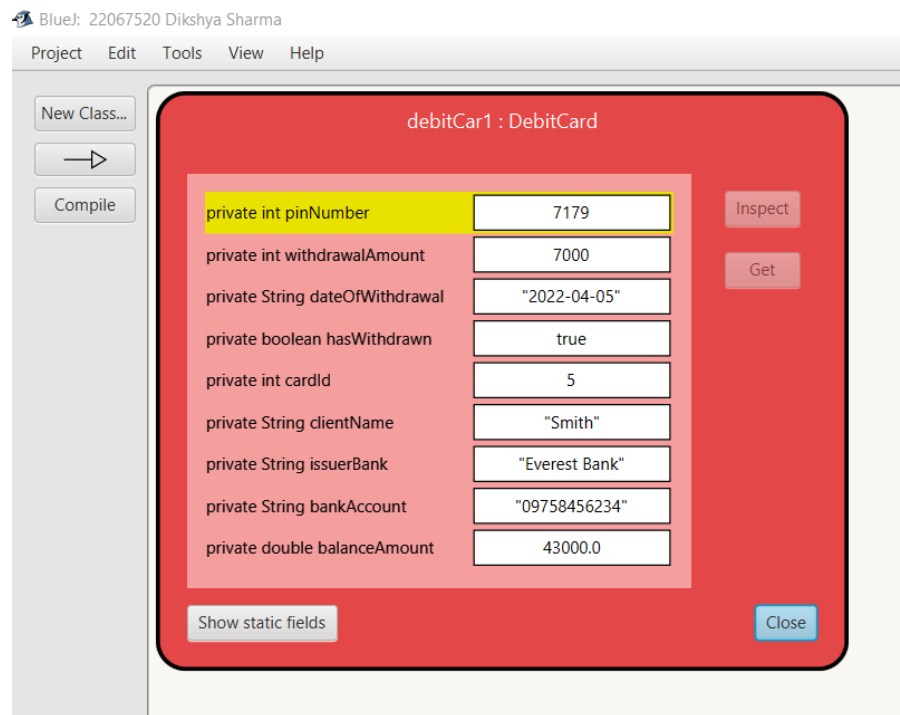


Figure 8: Inspecting the DebitCard Class after amount withdrawal

6.2 Test 2

Test No.	2
Objective:	To inspect the credit card class, set the credit limit and re-inspect the credit card class
Action:	<p>➔ The credit card is called with the following arguments:</p> <p>cardId : 6</p> <p>clientName : "John"</p> <p>issuerBank : "Everest Bank"</p> <p>bankAccount : "09758456234"</p> <p>balanceAmount : 80000</p> <p>cvcNumber : 745</p> <p>interestRate : 2.86</p> <p>expirationDate : "2023-11-23"</p> <p>➔ Inspection of the credit card class.</p> <p>➔ void setCreditLimit is called with the following arguments:</p> <p>creditLimit : 500000</p> <p>gracePeriod : 25</p> <p>➔ Re- inspection of the debit card class.</p>
Expected Result :	The credit limit would be set for client's bank account.
Actual Result :	The credit limit was set for client's bankaccount.
Conclusion:	The test is successful.

Table 2:Inspect the credit card class, set the credit limit and re-inspect the credit card class

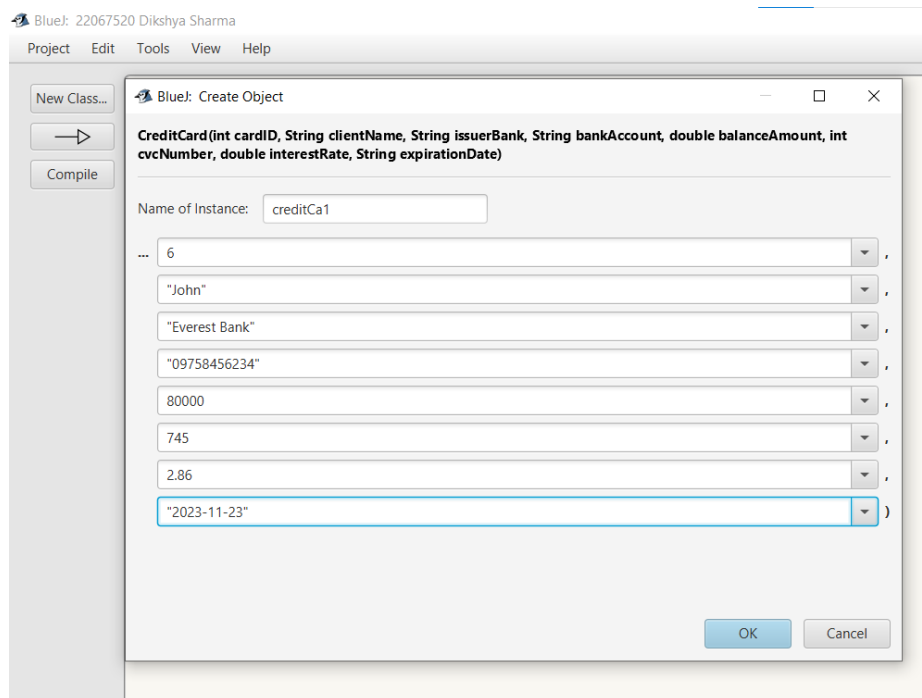


Figure 9: Inserting data in the attributes of CreditCard Class

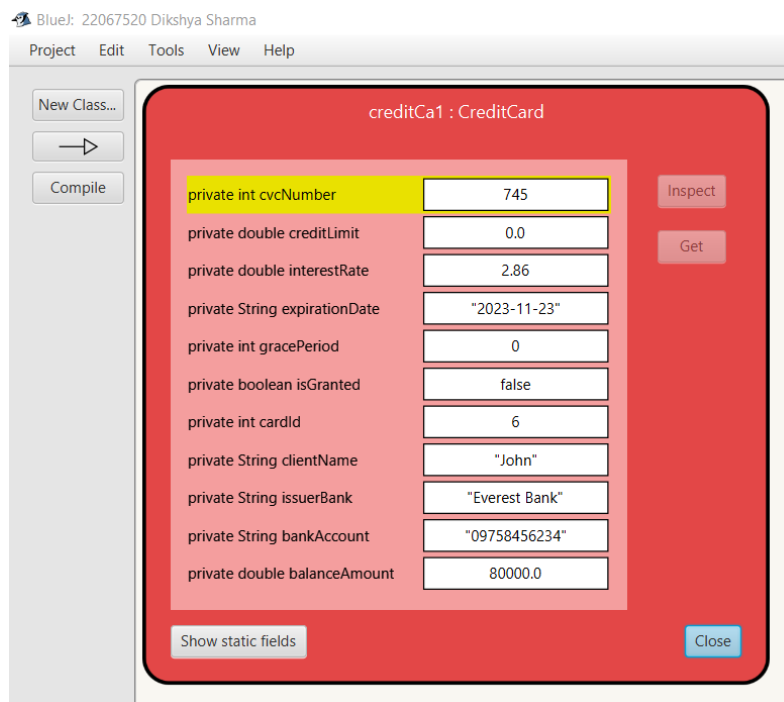


Figure 10: Inspecting DebitCard Class

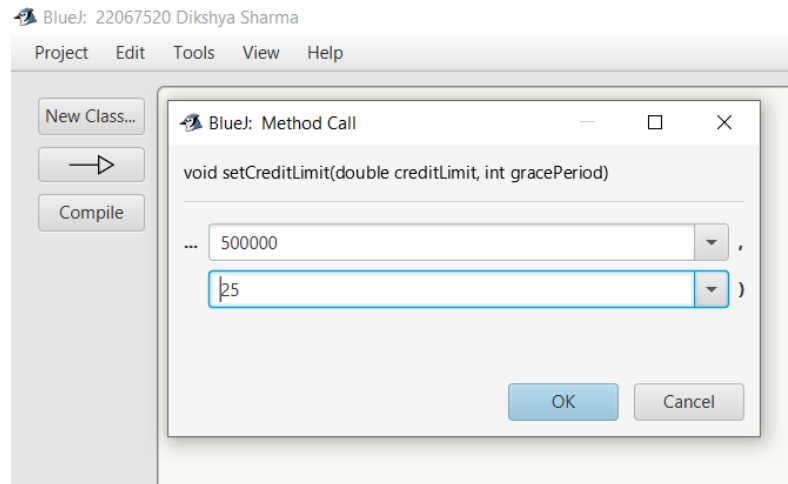


Figure 11: Setting the Credit Limit

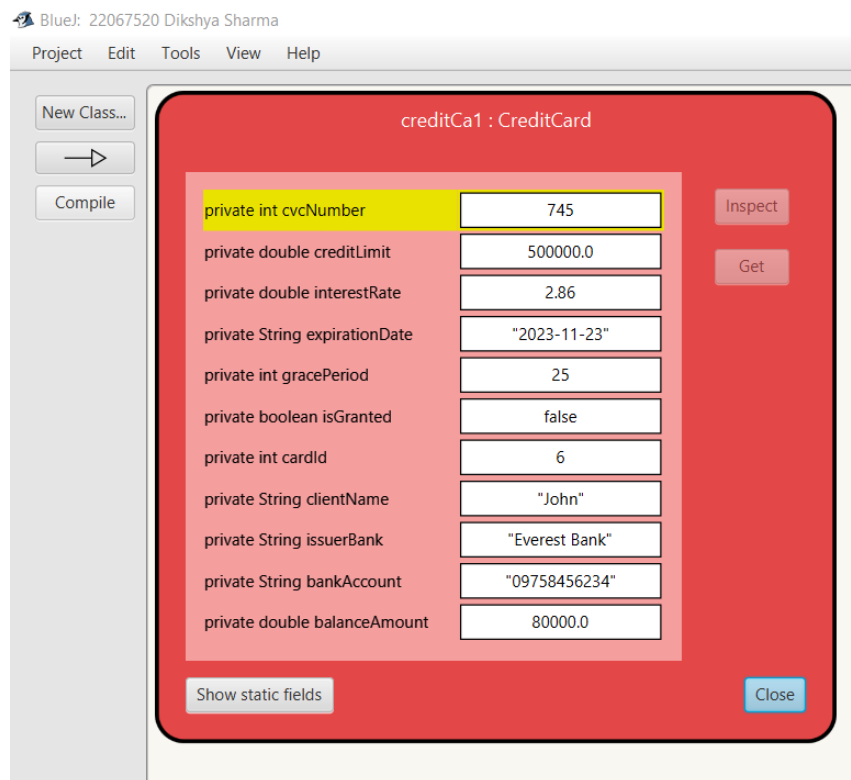


Figure 12: Inspecting CreditCard Class after setting credit limit

6.3 Test 3

Test No.	3
Objective:	To inspect the credit card class again after cancelling the credit card.
Action:	<p>➔ The credit card is called with the following arguments:</p> <p>cardId : 6</p> <p>clientName : "John"</p> <p>issuerBank : "Everest Bank"</p> <p>bankAccount : "09758456234"</p> <p>balanceAmount : 80000</p> <p>cvcNumber : 745</p> <p>interestRate : 2.86</p> <p>expirationDate : "2023-11-23"</p> <p>➔ Inspection of the credit card class.</p> <p>➔ void cancelCreditCard is called</p> <p>➔ Re- inspection of the credit card class.</p>
Expected Result :	The credit limit would be set for client's bank account.
Actual Result :	The credit limit was set for client's bankaccount.
Conclusion:	The test is successful.

Table 3:Inspect the credit card class again after cancelling the credit card

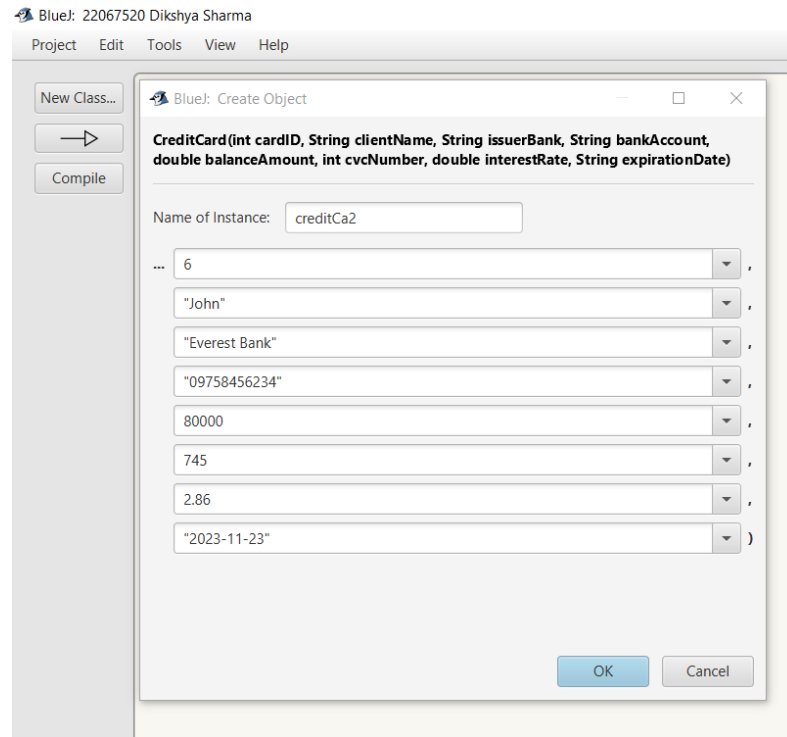


Figure 13: Inserting data in the attributes of CreditCard Class (Test 3)

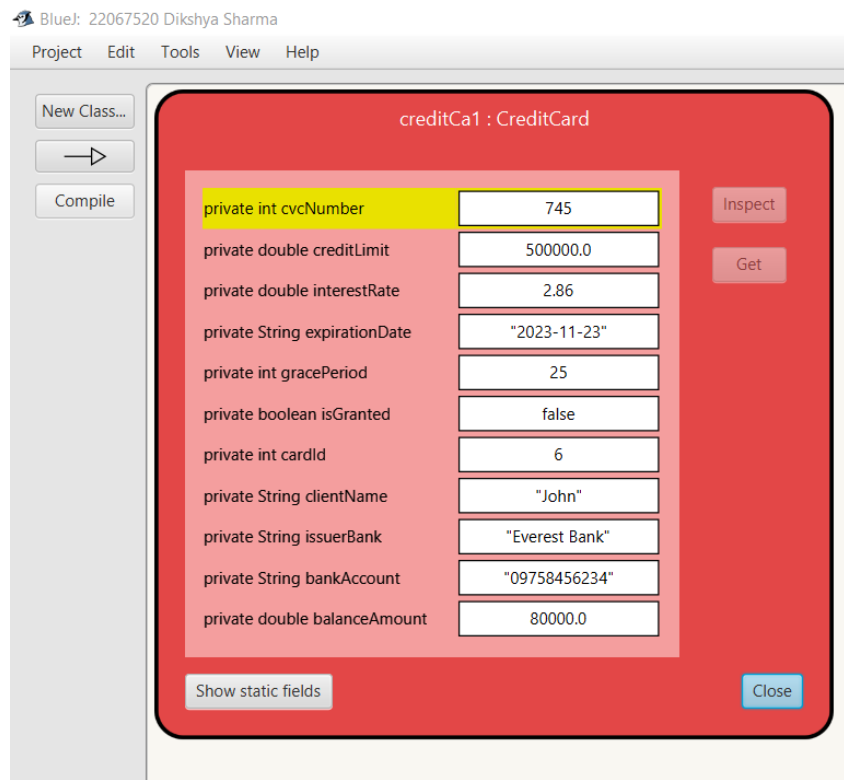


Figure 14: Inspecting CreditCard Class (Test 3)

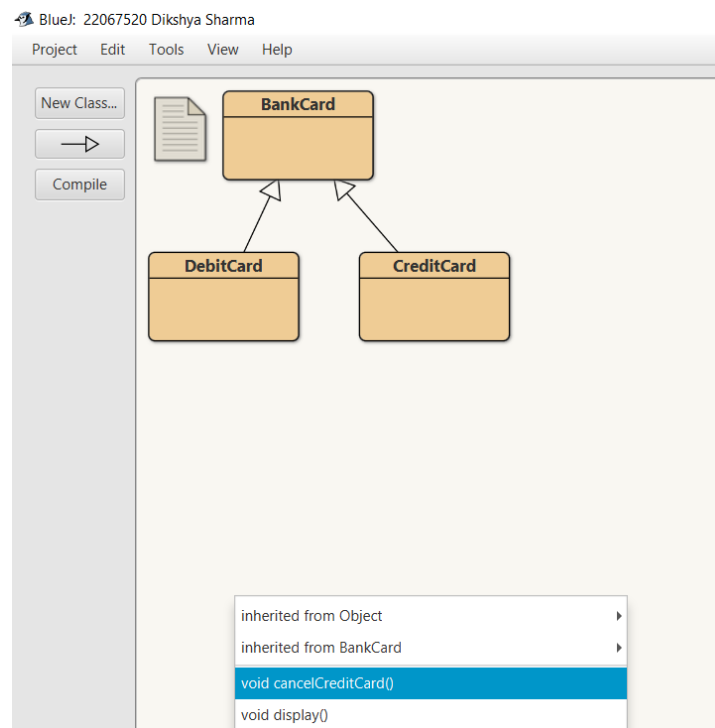


Figure 15: Cancelling Credit Card

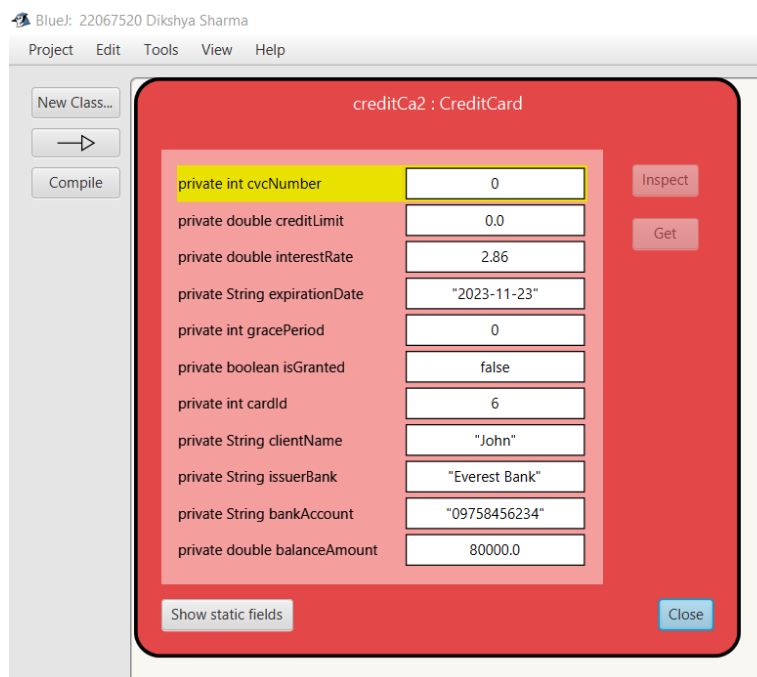


Figure 16: Inspecting CreditCard Class after cancelling credit card

6.4 Test 4

Test No.	4
Objective:	To display the details of Debit Card and Credit Card.
Action:	<p>➔ The debit card is called with the following arguments:</p> <p>balanceAmount : 50000</p> <p>cardId : 5</p> <p>bankAccount : "09758456234"</p> <p>issuerBank : "Everest Bank"</p> <p>clientName : "Smith"</p> <p>pinNumber : 7179</p> <p>➔ Inspection of debit card class</p> <p>➔ The credit card is called with the following arguments:</p> <p>cardId : 6</p> <p>clientName : "John"</p> <p>issuerBank : "Everest Bank"</p> <p>bankAccount : "09758456234"</p> <p>balanceAmount : 80000</p> <p>cvcNumber : 745</p> <p>interestRate : 2.86</p> <p>expirationDate : "2023-11-23"</p> <p>➔ Inspection of the credit card class.</p>
Expected Result :	Details of debit card and credit card would be displayed.
Actual Result :	Details of debit card and credit card was displayed.
Conclusion:	The test is successful.

Table 4:Display the details of debit card and credit card

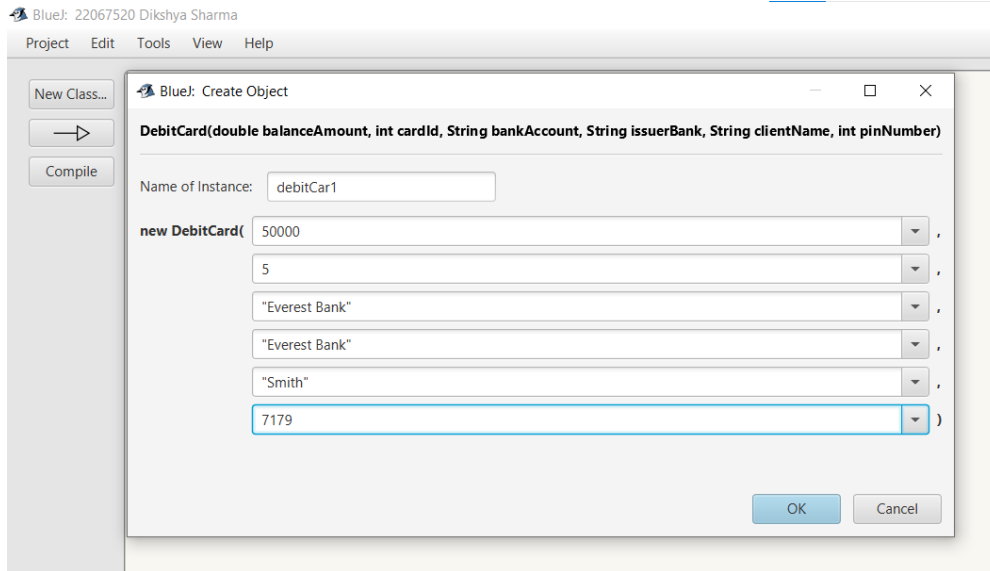


Figure 17: Inserting data in the attributes of DebitCard Class (Test 4)

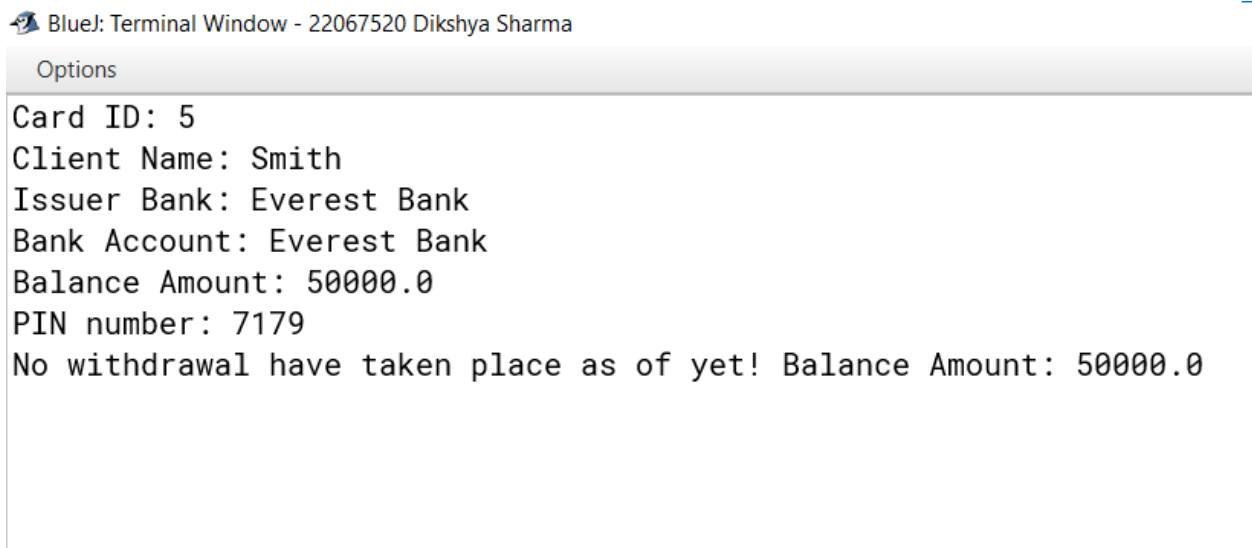


Figure 18: Displaying the DebitCard Class

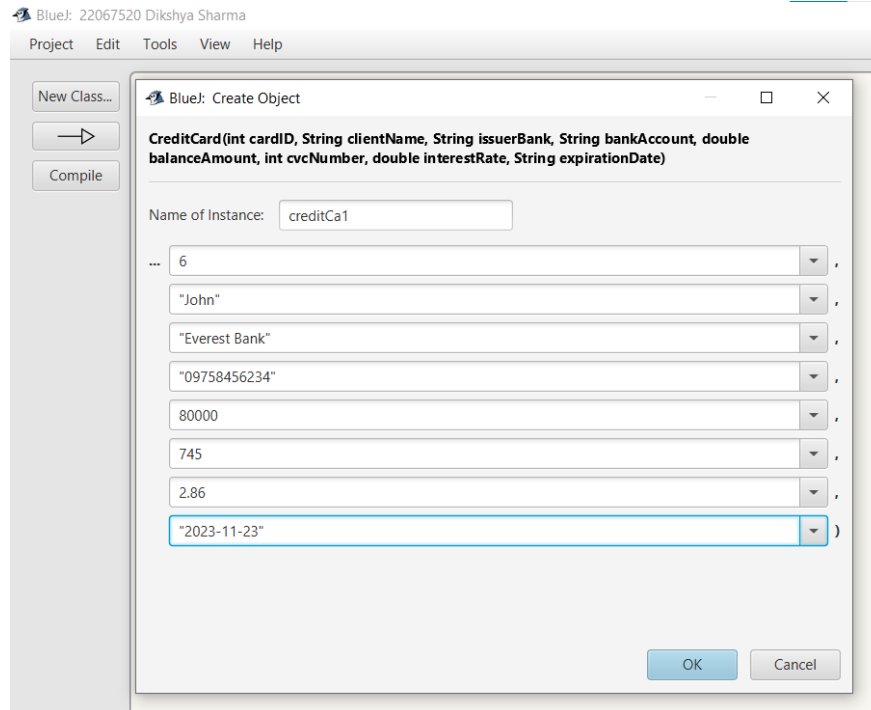


Figure 19: Inserting data in the attributes of CreditCard Class (Test 4)

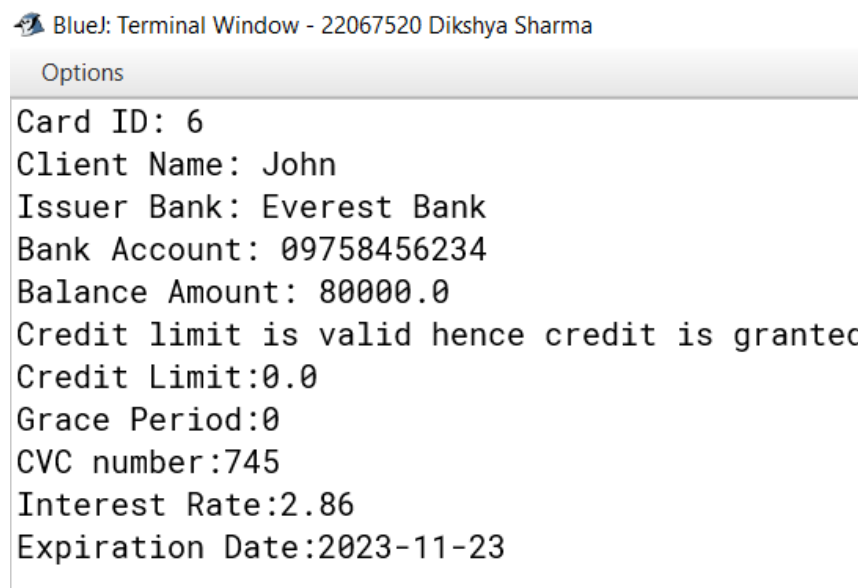


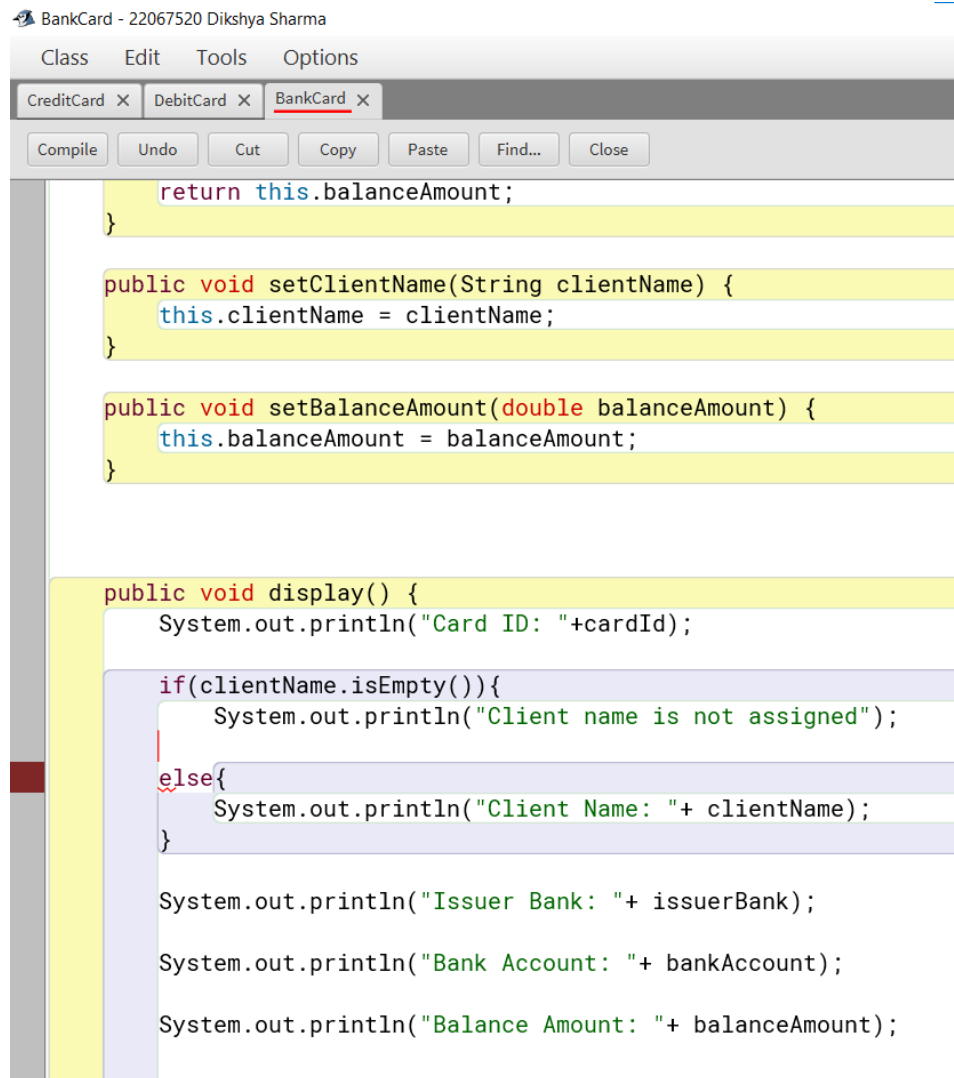
Figure 20: Displaying the DebitCard Class

7. ERROR DETECTION

The error encountered during this coursework are listed below:

7.1 Error 1:

In the display method in BankCard Class an error was encountered.

A screenshot of an IDE window titled 'BankCard - 22067520 Dikshya Sharma'. The window has tabs for 'CreditCard', 'DebitCard', and 'BankCard'. The 'BankCard' tab is active, showing a Java class with methods: 'return this.balanceAmount;', 'setClientName', 'setBalanceAmount', and 'display'. The 'display' method contains a conditional statement with an 'if' block and an 'else' block. A red squiggly line is under the 'else' keyword, indicating a syntax error. The code is as follows:

```
return this.balanceAmount;
}

public void setClientName(String clientName) {
    this.clientName = clientName;
}

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

public void display() {
    System.out.println("Card ID: "+cardId);

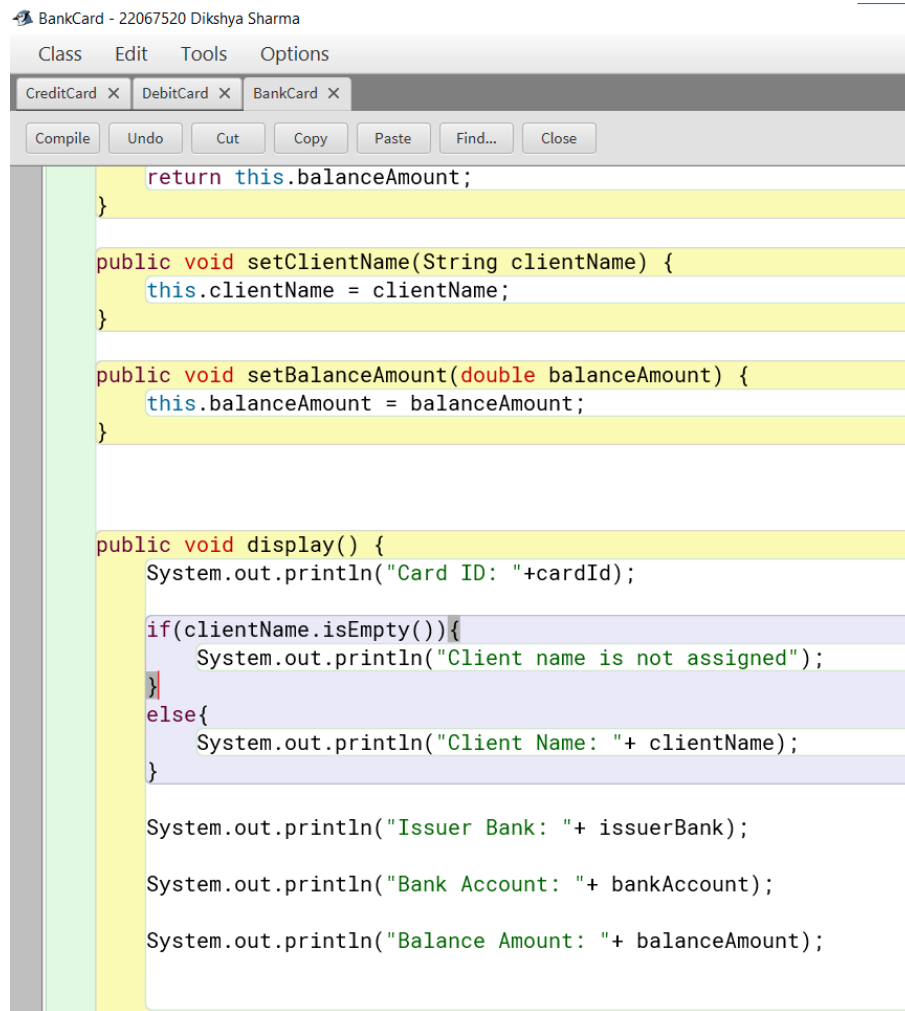
    if(clientName.isEmpty()){
        System.out.println("Client name is not assigned");
    }
    else{
        System.out.println("Client Name: "+ clientName);
    }

    System.out.println("Issuer Bank: "+ issuerBank);
    System.out.println("Bank Account: "+ bankAccount);
    System.out.println("Balance Amount: "+ balanceAmount);
}
```

Figure 21: Syntax Error

Correction Of Error 1:

The error was a syntax error where a closing curly brace was missing. The error was corrected by placing a closing curly brace.



```
BankCard - 22067520 Dikshya Sharma
Class Edit Tools Options
CreditCard X DebitCard X BankCard X
Compile Undo Cut Copy Paste Find... Close

return this.balanceAmount;
}

public void setClientName(String clientName) {
    this.clientName = clientName;
}

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

public void display() {
    System.out.println("Card ID: "+cardId);

    if(clientName.isEmpty()){
        System.out.println("Client name is not assigned");
    }
    else{
        System.out.println("Client Name: "+ clientName);
    }

    System.out.println("Issuer Bank: "+ issuerBank);

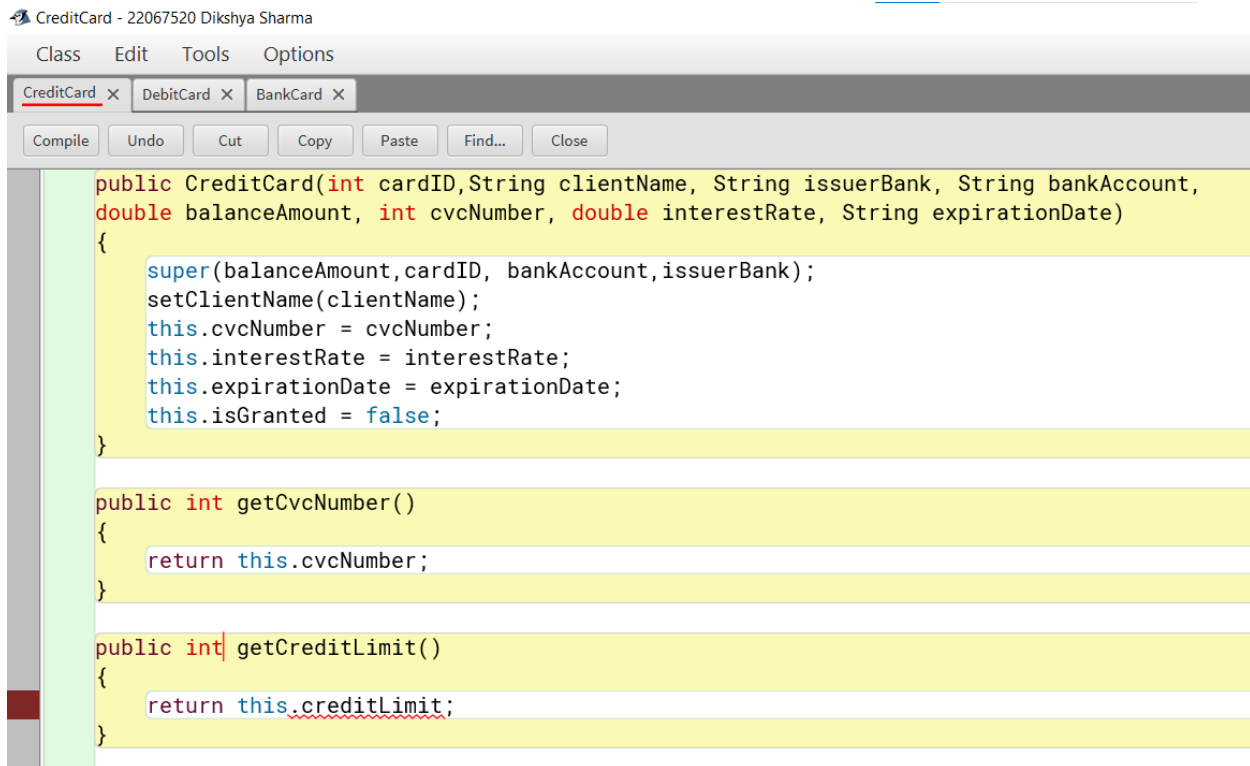
    System.out.println("Bank Account: "+ bankAccount);

    System.out.println("Balance Amount: "+ balanceAmount);
}
```

Figure 22:Correction of Syntax Error

7.2 Error 2:

An incompatible type error was encountered on CreditCard Class in an accessor method.



```
public CreditCard(int cardID, String clientName, String issuerBank, String bankAccount,
double balanceAmount, int cvcNumber, double interestRate, String expirationDate)
{
    super(balanceAmount, cardID, bankAccount, issuerBank);
    setClientName(clientName);
    this.cvcNumber = cvcNumber;
    this.interestRate = interestRate;
    this.expirationDate = expirationDate;
    this.isGranted = false;
}

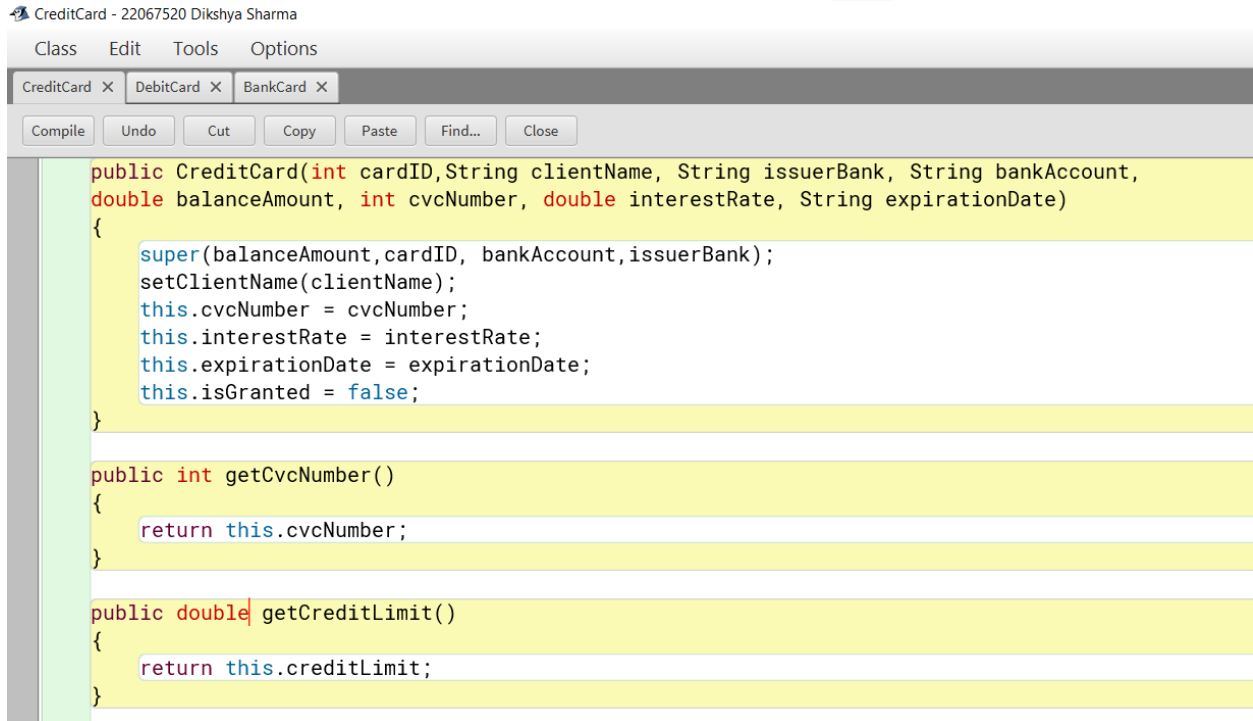
public int getCvcNumber()
{
    return this.cvcNumber;
}

public int| getCreditLimit()
{
    return this.creditLimit;
}
```

Figure 23: Semantic Error

Correction Of Error 2:

An semantic error was found where integer data type was used instead of double as creditLimit attribute is set as double data type. It was corrected by replacing integer data type with double.



The screenshot shows an IDE window titled "CreditCard - 22067520 Dikshya Sharma". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar are tabs for "CreditCard X", "DebitCard X", and "BankCard X". A toolbar contains buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The main editor area displays the following Java code for the CreditCard class:

```
public CreditCard(int cardID,String clientName, String issuerBank, String bankAccount,
double balanceAmount, int cvcNumber, double interestRate, String expirationDate)
{
    super(balanceAmount,cardID, bankAccount,issuerBank);
    setClientName(clientName);
    this.cvcNumber = cvcNumber;
    this.interestRate = interestRate;
    this.expirationDate = expirationDate;
    this.isGranted = false;
}

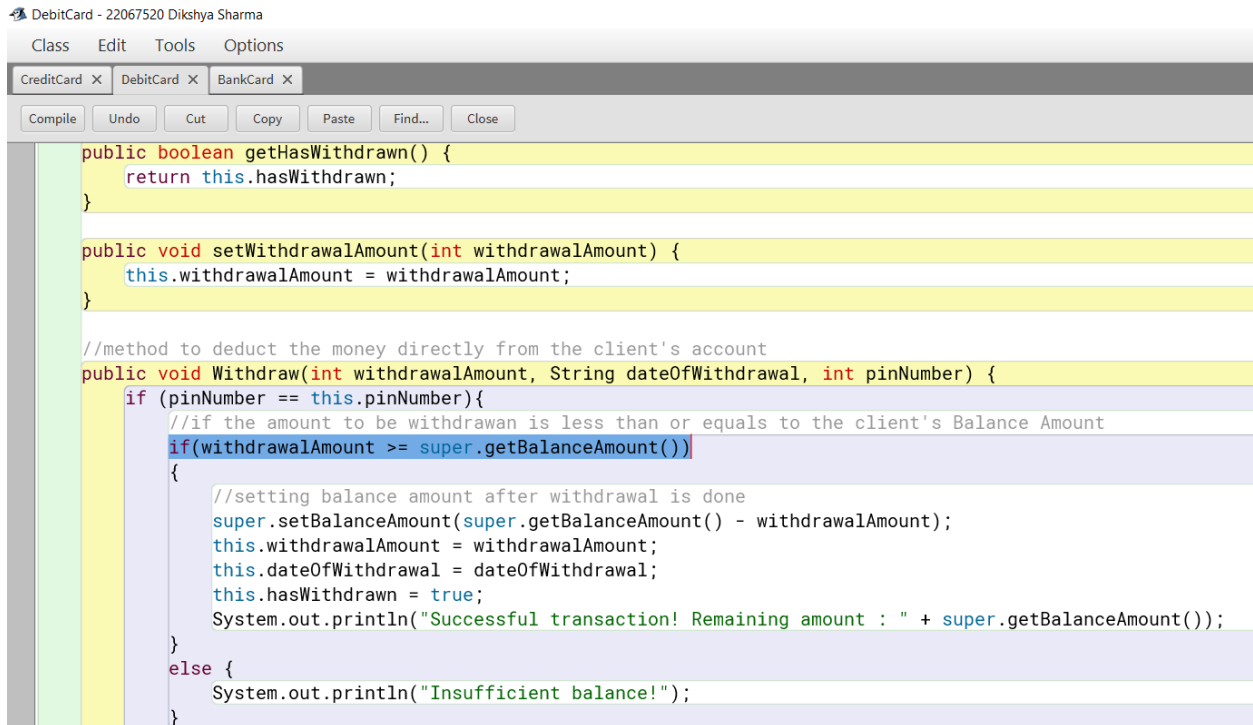
public int getCvcNumber()
{
    return this.cvcNumber;
}

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

Figure 24:Correction of Semantic Error

7.3 Error 3:

An logical error was encountered in DebitCard Class on Withdraw method where instead of smaller than sign a greater than sign was used which the compiler didn't show but it affected the end result of the program.



```
DebitCard - 22067520 Dikshya Sharma
Class Edit Tools Options
CreditCard X DebitCard X BankCard X
Compile Undo Cut Copy Paste Find... Close

public boolean getHasWithdrawn() {
    return this.hasWithdrawn;
}

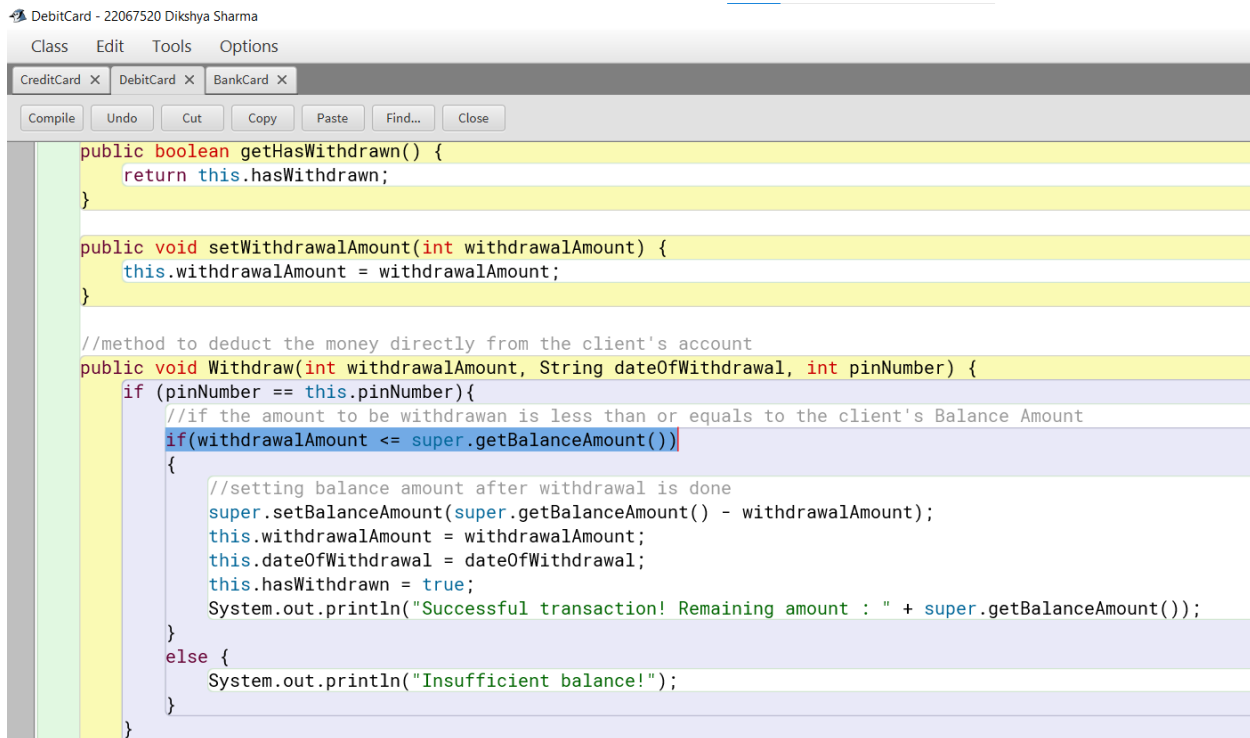
public void setWithdrawalAmount(int withdrawalAmount) {
    this.withdrawalAmount = withdrawalAmount;
}

//method to deduct the money directly from the client's account
public void Withdraw(int withdrawalAmount, String dateOfWithdrawal, int pinNumber) {
    if (pinNumber == this.pinNumber){
        //if the amount to be withdrawn is less than or equals to the client's Balance Amount
        if(withdrawalAmount >= super.getBalanceAmount())
        {
            //setting balance amount after withdrawal is done
            super.setBalanceAmount(super.getBalanceAmount() - withdrawalAmount);
            this.withdrawalAmount = withdrawalAmount;
            this.dateOfWithdrawal = dateOfWithdrawal;
            this.hasWithdrawn = true;
            System.out.println("Successful transaction! Remaining amount : " + super.getBalanceAmount());
        }
        else {
            System.out.println("Insufficient balance!");
        }
    }
}
```

Figure 25: Logical Error

Correction Of Error 3:

It was a logical error which was encountered while running a program and was solved by using less than sign which then meets the condition given in the question.



```
DebitCard - 22067520 Dikshya Sharma
Class Edit Tools Options
CreditCard X DebitCard X BankCard X
Compile Undo Cut Copy Paste Find... Close

public boolean getHasWithdrawn() {
    return this.hasWithdrawn;
}

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

//method to deduct the money directly from the client's account
public void Withdraw(int withdrawalAmount, String dateOfWithdrawal, int pinNumber) {
    if (pinNumber == this.pinNumber){
        //if the amount to be withdrawn is less than or equals to the client's Balance Amount
        if(withdrawalAmount <= super.getBalanceAmount())
        {
            //setting balance amount after withdrawal is done
            super.setBalanceAmount(super.getBalanceAmount() - withdrawalAmount);
            this.withdrawalAmount = withdrawalAmount;
            this.dateOfWithdrawal = dateOfWithdrawal;
            this.hasWithdrawn = true;
            System.out.println("Successful transaction! Remaining amount : " + super.getBalanceAmount());
        }
        else {
            System.out.println("Insufficient balance!");
        }
    }
}
```

Figure 26:Correction of Logical Error

8. CONCLUSION

The coursework given was the first coursework for this module which was certainly a challenging task to do. As writing program is all about creativity and dedication, it took me a whole lot of commitment and effort to make this project a success. The tutorial sessions, workshop classes and the learning materials that were offered were a great boost to the knowledge for this project. Besides this I also took some help from the Internet. The use of Google Search really aided in overcoming obstacles and resolving issues, which undoubtedly improved my research skills.

I well learned the concept of Object-Oriented Programming very clearly as I did a lot of research to better understand the ideas behind constructors, accessors and mutators while writing this program. I also learned the better way of writing a program which includes use of comments, proper way of indentation and so much more. I had a moment where I thought I would not finish this coursework on time as it was my first time writing a report, but lecturers and tutors were incredibly helpful that made me complete this coursework on time. Despite all the challenges, I enjoyed doing this coursework to the fullest. It was indeed a great learning experience.

References

Anon., n.d. *Computer Hope*. [Online]
Available at: <https://www.computerhope.com/jargon/m/microsoft-word.htm>
[Accessed 11 06 2021].

Anon., n.d. *Computer Hope*. [Online]
Available at: <https://www.computerhope.com/jargon/d/drawio.htm>
[Accessed 06 02 2020].

Kolling, M., B. Q., Patterson, A. & RosenBerg, J., 2010. The BlueJ System and its Pedagogy. *Computer Science Education*, 13(4), p. 3.

APPENDIX

a. For BankCard Class

```
/**
```

```
 * Write a description of class BankCard here.
```

```
 *
```

```
 * @author (22067520 Dikshya Sharma)
```

```
 * @version (1.0.0)
```

```
 */
```

```
public class BankCard
```

```
{
```

```
    private int cardId;
```

```
    private String clientName;
```

```
    private String issuerBank;
```

```
    private String bankAccount;
```

```
    private double balanceAmount;
```

```
    public BankCard(double balanceAmount, int cardId, String bankAccount,String  
    issuerBank)
```

```
    {
```

```
        this.balanceAmount = balanceAmount;

        this.cardId = cardId;

        this.bankAccount = bankAccount;

        this.issuerBank = issuerBank;

        this.clientName = "";

    }
```

```
    public int getCardId(){

        return this.cardId;

    }
```

```
    public String getClientName() {

        return this.clientName;

    }
```

```
    public String getIssuerBank() {

        return this.issuerBank;

    }
```

```
    public String getBankAccount() {

        return this.bankAccount;

    }
```

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

```
public void setClientName(String clientName) {  
    this.clientName = clientName;  
}
```

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

```
public void display() {  
    System.out.println("Card ID: "+cardId);  
  
    if(clientName.isEmpty()){  
        System.out.println("Client name is not assigned");  
    }  
    else{  
        System.out.println("Client Name: "+ clientName);  
    }  
}
```

```
System.out.println("Issuer Bank: "+ issuerBank);
```

```
System.out.println("Bank Account: "+ bankAccount);
```

```
System.out.println("Balance Amount: "+ balanceAmount);
```

```
}
```

```
}
```

b. For DebitCard Class

```
/**
```

```
 * Write a description of class BankCard here.
```

```
 *
```

```
 * @author (22067520 Dikshya Sharma)
```

```
 * @version (1.0.0)
```

```
 */
```

```
public class DebitCard extends BankCard
```

```
{
```

```
    private int pinNumber;
```

```
    private int withdrawalAmount;
```

```
    private String dateOfWithdrawal;
```

```
    private boolean hasWithdrawn;
```

```
    public DebitCard(double balanceAmount, int cardId, String bankAccount, String  
issuerBank, String clientName, int pinNumber) {
```

```
        super(balanceAmount, cardId, bankAccount, issuerBank);
```

```
        super.setClientName(clientName);
```

```
        this.pinNumber = pinNumber;
```

```
        this.hasWithdrawn = false;
```

```
    }
```

```
public int getPinNumber() {  
    return this.pinNumber;  
}
```

```
public int getWithdrawalAmount() {  
    return this.withdrawalAmount;  
}
```

```
public String getDateOfWithdrawal() {  
    return this.dateOfWithdrawal;  
}
```

```
public boolean getHasWithdrawn() {  
    return this.hasWithdrawn;  
}
```

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

```
//method to deduct the money directly from the client's account
```

```
public void Withdraw(int withdrawalAmount, String dateOfWithdrawal, int pinNumber)
{
    if (pinNumber == this.pinNumber){

        //if the amount to be withdrawan is less than or equals to the client's Balance
        Amount

        if(withdrawalAmount <= super.getBalanceAmount())
        {

            //setting balance amount after withdrawal is done

            super.setBalanceAmount(super.getBalanceAmount() - withdrawalAmount);

            this.withdrawalAmount = withdrawalAmount;

            this.dateOfWithdrawal = dateOfWithdrawal;

            this.hasWithdrawn = true;

            System.out.println("Successful transaction! Remaining amount : " +
super.getBalanceAmount());

        }

        else {

            System.out.println("Insufficient balance!");

        }

    }

    else {

        System.out.println("Invalid pin number!");

    }

}
```



```
}

public void display() {
    super.display();
    System.out.println("PIN number: " + pinNumber);

    if (hasWithdrawn == true) {
        System.out.println("Withdrawal amount: " + withdrawalAmount);
        System.out.println("Date of withdrawal: " + dateOfWithdrawal);
    }
    else{
        System.out.println("No withdrawal have taken place as of yet! Balance Amount:
"+ super.getBalanceAmount());
    }
}
}
```

c. For CreditCard Class

```
/**
```

```
 * Write a description of class CreditCard here.
```

```
 *
```

```
 * @author (22067520 Dikshya Sharma)
```

```
 * @version (1.0.0)
```

```
 */
```

```
public class CreditCard extends BankCard
```

```
{
```

```
    private int cvcNumber;
```

```
    private double creditLimit;
```

```
    private double interestRate;
```

```
    private String expirationDate;
```

```
    private int gracePeriod;
```

```
    private boolean isGranted;
```

```
    public CreditCard(int cardID,String clientName, String issuerBank, String  
bankAccount,
```

```
double balanceAmount, int cvcNumber, double interestRate, String expirationDate)
```

```
{
```

```
    super(balanceAmount,cardID, bankAccount,issuerBank);
```

```
    super.setClientName(clientName);
```

```
    this.cvcNumber = cvcNumber;

    this.interestRate = interestRate;

    this.expirationDate = expirationDate;

    this.isGranted = false;
}
```

```
public int getCvcNumber()
{
    return this.cvcNumber;
}
```

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

```
public double getInterestRate()
{
    return this.interestRate;
}
```

```
public String getExpirationDate()
{

```

```
        return this.expirationDate;
    }

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

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

    public void setCreditLimit(double creditLimit, int gracePeriod)
    {
        this.creditLimit = creditLimit;
        this.gracePeriod = gracePeriod;
        if(creditLimit <= 2.5 * (super.getBalanceAmount())){
            System.out.println("Credit is successfully granted!");
            this.isGranted = true;
        }else{
            System.out.println("Credit couldn't be issued due to a credit limit overuse");
        }
    }
```

```
}
```

```
public void cancelCreditCard()
```

```
{
```

```
    this.cvcNumber = 0;
```

```
    this.creditLimit = 0;
```

```
    this.gracePeriod = 0;
```

```
    this.isGranted = false;
```

```
}
```

```
public void display()
```

```
{
```

```
    super.display();
```

```
    if(isGranted == true){
```

```
        System.out.println("Credit limit is valid hence credit is granted");
```

```
        System.out.println("Credit Limit:" + this.creditLimit);
```

```
        System.out.println("Grace Period:" + this.gracePeriod);
```

```
    }
```

```
    else{
```

```
        System.out.println("Credit is not granted.");
```

```
    }
```

```
    System.out.println("CVC number:" + this.cvcNumber);
```

```
    System.out.println("Interest Rate:" + this.interestRate);
```

```
        System.out.println("Expiration Date:" + this.expirationDate);  
    }  
}
```