Module Code & Module Title CS4001NP
Introduction to Programming

Assessment weightage & Type 30% Individual Coursework - 1

Year and Semester2022-23 Autumn

Student name: Prabal Gurung
Group: C3
London met ID: 22069041
College ID: NP04CP4A220088
Assignment due date: January 27, 2023
Assignment submission date: January 27, 2023

Declaration:

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

Contents

List Of Figures:

List Of Tables:

1. Introduction

The following project was handed to us to determine our ability against the real-world problem with Object-oriented concepts. The specified project is delivered in BlueJ where only three classes are represented. One class is super-class i.e., bank card and the remaining two are sub-class i.e., Credit Card and Bank Card. The following program is coded in programming interface named blueJ. This interface has it's own in-built computer which helps us to help with the coding and run the app a lot easier.

An object-oriented concept in java is based on declaring classes, creating objects from them, and interacting between these objects. This concept consists of four fundamentals, and we 2will be using one of them i.e., Encapsulation (data hiding). And one of the important pillars of the object-oriented concept, Inheritance. Inheritance is a mechanism by which one class helps another class to inherit its feature from any other class that we have determined.

In this report, we will discuss the project in various ways; class diagram, pseudocode, main code with the test, and errors that I have encountered during the project.

2. Class Diagram

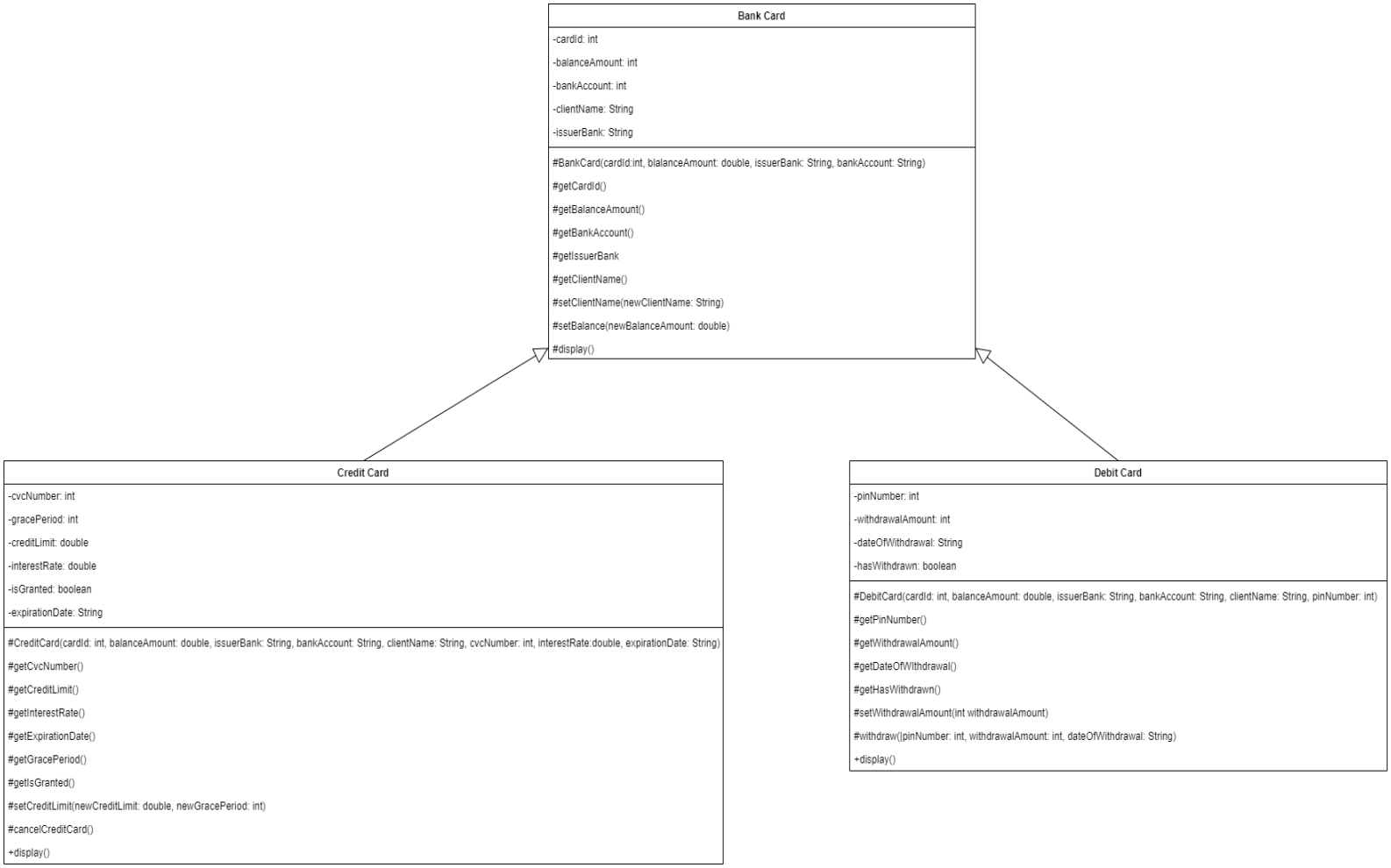

Figure 1 Class diagram

2.1 Bank Card Class Diagram

Bank Card
-cardId: int -balanceAmount: int -bankAccount: int -clientName: String -issuerBank: String
#BankCard(cardId:int, blalanceAmount: double, issuerBank: String, bankAccount: String) #getCardId() #getBalanceAmount() #getBankAccount() #getIssuerBank #getClientName() #setClientName(newClientName: String) #setBalance(newBalanceAmount: double) #display()

Figure 2 Bank Card Class Diagram

2.2 Credit Card Class Diagram

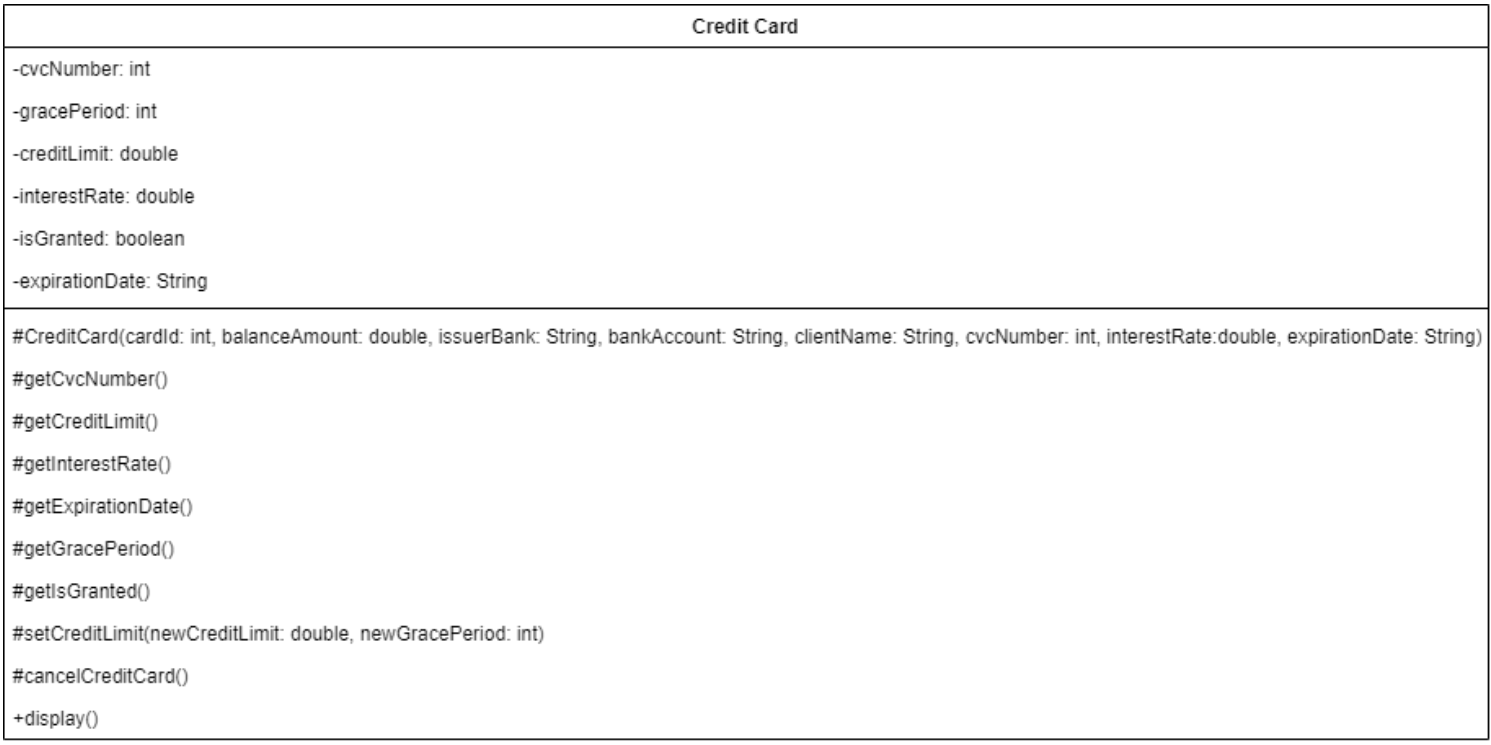

Figure 3 Credit Card Class Diagram

2.3 Debit Class Diagram

Debit Card
-pinNumber: int -withdrawalAmount: int -dateOfWithdrawal: String -hasWithdrawn: boolean
#DebitCard(cardId: int, balanceAmount: double, issuerBank: String, bankAccount: String, clientName: String, pinNumber: int) #getPinNumber() #getWithdrawalAmount() #getDateOfWIthdrawal() #getHasWithdrawn() #setWithdrawalAmount(int withdrawalAmount) #withdraw(|pinNumber: int, withdrawalAmount: int, dateOfWithdrawal: String) +display()

Figure 4 Debit Card Class Diagram

3. Pseudocode

3.1 Pseudocode for BankCard

```
DECLARE cardId
DECLARE balanceAmount
DECLARE bankAccount
DECLARE issuerBank
DECLARE clientName

NO ARGUEMENT CONSTRUCTOR
END CONSTRUCTOR

PARAMETARIZED CONSTRUCTOR(cardId, balanceAmount, issuerBank, bankAccount)
        INITIALIZE cardId
        INITIALIZE balanceAmount
        INITIALIZE bankAccount
        INITIALIZE issuerBank
        INITIALIZE clientName
END CONSTRUCTOR

FUNCTION getCardId
        RETURN cardId
END FUNCTION

FUNCTION getBalanceAmount
        RETURN balanceAmount
END FUNCTION

FUNCTION getIssuerBank
        RETURN issuerBank
END FUNCTION

FUNCTION getClientName
        RETURN clientName
END FUNCTION

FUNCTION setClientName (newClientName)
        INITIALIZE newClientName
END FUNCTION

FUNCTION setBalance(newBalanceAmount)
        newBalanceAmount
END FUNCTION

FUNCTION display
        PRINT (cardId, balanceAmount, issuerBank, bankAccount)
```

```
IF clientName != " "
        PRINT (clientName)
    ELSE
        PRINT (clientName not assigned)
    END IF
END FUNCTION
END
```

3.2 Pseudocode for DebitCard

```
INITIALIZE pinNumber
INITIALIZE withdrawalAmount
INITIALIZE dateOfWithdrawal
INITIALIZE hasWithdrawn

PARAMETARIZED CONSTRUCTOR DeditCard (cardId, balanceAmount, issuerBank,
bankAcount, clientName, pinNumber)
        CALL (cardId, balanceAmount, issuerBank, bankAccount)
        INITIALIZE pinNumber
        INITIALIZE hasWithdrawn
END CINSTRUCTOR

FUNCTION getPinNumber
        RETURN pinNumber
END FUNCTION

FUNCTION getWithdrawalAmount
        RETURN withdrawalAmount
END FUNCTION

FUNCTION getDateOfWithdrawal
        RETURN dateOfWithdrawal
END FUNCTION

FUNCTION getHasWithdrawn
        RETURN hasWithdrawn
END FUNCTION

FUNCTION setWithdrawalAmount(withdrawalAmount)
        RETURN withdrawalAmount
END FUNCTION

FUNCTION withdraw (pinNumber, withdrawalAmount, dateOfWithdrawal)
        IF (pinNumber != 0)
                IF(super.getBalanceAmount() >= withdrawalAmount)
                  SET hasWithdrawn = True
                  SET dateOfWithdrawal = dateOfWithdrawal
                  SETsetBalanceAmount(getBalanceAmount() – withdrawalAmount)
                ELSE
                        PRINT (Insufficient balance)
        END IF
ELSE
                PRINT (Invalid pin number)


        END IF
```

```
END FUNCTION

        FUNCTION display
                CALL display
                PRINT pinNumber
                IF (hasWithdrawn = true)
                        PRINT (withdrawalAmount, dateOfWithdrawal)
                ELSE
                        PRINT (super.getBalanceAmount)
                END IF
        END FUNCTION
END
```

3.3 Pseudocode for CreditCard

```
    INITIALIZE cvcNumber
    INITIALIZE creditLimit
    INITIALIZE interestRate
    INITIALIZE expirationDate
    INITIALIZE gracePeriod
    INITIALIZE isGranted

PARAMETERIZED CONSTRUCTOR (cardId, balanceAmount. issuerBank, bankAccount,
clientName, cvcNumber, interestRate, expirationDate)
        CALL (cardId, balanceAmount, issuerBank, bankAccount)
        INITIALIZE cvcNumber
        INITIALIZE interestRate
        INITIALIZE expirationDate
        INITIALIZE expirationDate
        INITIALIZE isGranted
    END CONSTRUCTOR

    FUNCTION getCvcNumber
        RETURN CvcNumber
    END FUNCTION

    FUNCTION getCreditLimit
        RETURN creditLimit
    END FUNCTION

    FUNCTION getInterestRate
        RETURN interestRate
    END FUNCTION

    FUNCTION getExpirationDate
        RETURN expirationDate
    END FUNCTION

    FUNCTION getGracePeriod
        RETURN gracePeriod
    END FUNCTION

    FUNCTION getIsGranted
        RETURN isGranted
    END FUNCTION

    FUNCTION setCreditLimit (newCreditLimit, newGracePeriod)
        IF (newCreditLimit <= (2.5 * super.getBalanceAmount()))
            INITIALIZE newCreditLimit
            INITIALIZE newGracePeriod
            SET isGranted = True
```

```
        ELSE
                PRINT (error)
        END IF
    END FUNCTION
    FUNCTION cancelCreditCard
        SET cvcNumber = 0
        SET creditLimit = 0
        SET gracePeriod = 0
        SET isGranted = false
    END FUNCTION

    FUNCTION display
        IF(isGranted = true)
                CALL.display()
                PRINT(cvcNumber, creditLimit, interestRate, expirationDate, gracePeriod)
        ELSE
    CALL.display()
                PRINT(cvcNumber, interestRate, expirationDate)
        ENDIF
    END FUNCTION
END
```

4. Description

Java is a multi-platform and network-centric programming language, which is mainly used for developing android apps and enterprise software. Java is a highly used platform in programming because of its higher cross-functionality and probability.

In java programming language, class is determined by a lot of things like attributes and methods. Focusing on method, it is like a function that exposes the behavior of an object. The required elements in the method declaration are the modifiers, return type, name, parameter list in parenthesis, and method body enclosed between the braces {}.

Method helps in the code reusability and optimizes the codes in itself. Basically, the method has a unique name within its class but can also have the same name as the other methods due to overloading.

4.1 Method of BankCard

BankCard
This is a no-argument constructor and pass parameters to sub-class.

BankCard (cardId, balanceAmount, issuerBank, bankAccount)
This is parameterized constructor which takes four parameters: balanceAmount, issuerBank, bankAccount.

getCardId
This method returns the integer value of cardId.

getBankAccount
This method returns the String value of bankAccount.

getIssuerBank
This method returns the String value of issuerBank.

getClientName
This method returns the String value of clientName.

setBalance(newBalanceAmount)
This method initializes newBalanceAmount value to balanceAmount

display()
This method prints (cardId, balanceAmount, issuerBank, bankAccount) after checking if client name is empty or not suitable message is displayed.

4.2 Method of DebitCard

DebitCard(cardId, balanceAmount, issuerBank, bankAccount, clientName, pinNumber
This is parameterized constructor. It calls (cardId, balanceAmount, issuerBank, bankAccount, clientName) from its parent class, initializes pinNumber and sets hasWithdrawn to False.

getPinNumber
This method returns the integer value of pinNumber

getWithdrawalAmount
This method returns the integer value of withdrawalAmount

getDateOfWithdrawal
This method returns the String value of dateOfWithdrawal

getHasWithdrawn
This method returns the boolean value of hasWithdrawn

setWithdrawalAmount (withdrawalAmount)
This method takes a new variable salary as a parameter and sets the value of instance variable salary.

withdraw(pinNumber, withdrawalAmount, dateOfWithdrawal)
This method whether the entered pin and given pin is same or not and checks again if super.getBalanceAmount >= withdrawalAmount and if all condition is true then hasWithdrawn is set to true, dateOfWithdrawal is set and super.setBalanceAmount is decreased with withdrawalAmount and if the condition is not met then a suitable message is shown.

display()
This method prints detail from parent class and also checks if hasWithdrawn is set to true or not, if it its true then pinNumber, withdrawalAmount and dateOfWithdrawal is printed else super.getBalanceAmount is printed.

4.3 Method of CreditCard

CreditCard(cardId, balanceAmount, issuerBank, bankAccount, clientName, cvcNumber, interestRate, expirationDate)
This method calls (cardId, balanceAmount, issuerBank, bankAccount) from super-class. It also initializes cvcNumber, interestRate, expirationDate and sets isGranted to false.

getCvcNumber()
This method returns the integer value of cvcNumber.

getCreditLimit()
This method returns the double value of creditLimit.

getInterestRate()
This method returns the double value of intesrestRate.

getExpirationDate()
This method returns the String value of expirationDate.

getGracePeriod()
This method returns the integer value of gracePeriod

getIsGranted()
This method returns the boolean value of isGranted.

setCreditLimit(newCreditLimit, NewGracePeriod)
This method checks whether the newCreditLimit is lower than or equal to super.getBalanceAmount * 2.5. If the limit is true then creditLimit, gracePeriod gets newCreditLimit, newGracePeriod respectively and isGranted is set to true else error message is shown.

cancelCreditCard()
sets cvcNumber, creditLimit, gracePeriod to 0 and isGranted to false.

display()
This method first checlks whether isGranted is true or flase. In case of true cvcNumber, creditLImit, interestRate, expiratioDate, and gracePeriod is printed else error message is shown.

5 Testing

After thoroughly examining and writing my program, I had to execute my program and test it myself whether its working properly or not. In this phase I will be showing all the test I have done during my time with the project.

5.1 Test 1

In test 1, According to question we are to Inspect the Debit Card class, withdraw the amount, and re-inspect the Debit Card Class.

Inspecting Debit Card Class:

Expected Output:

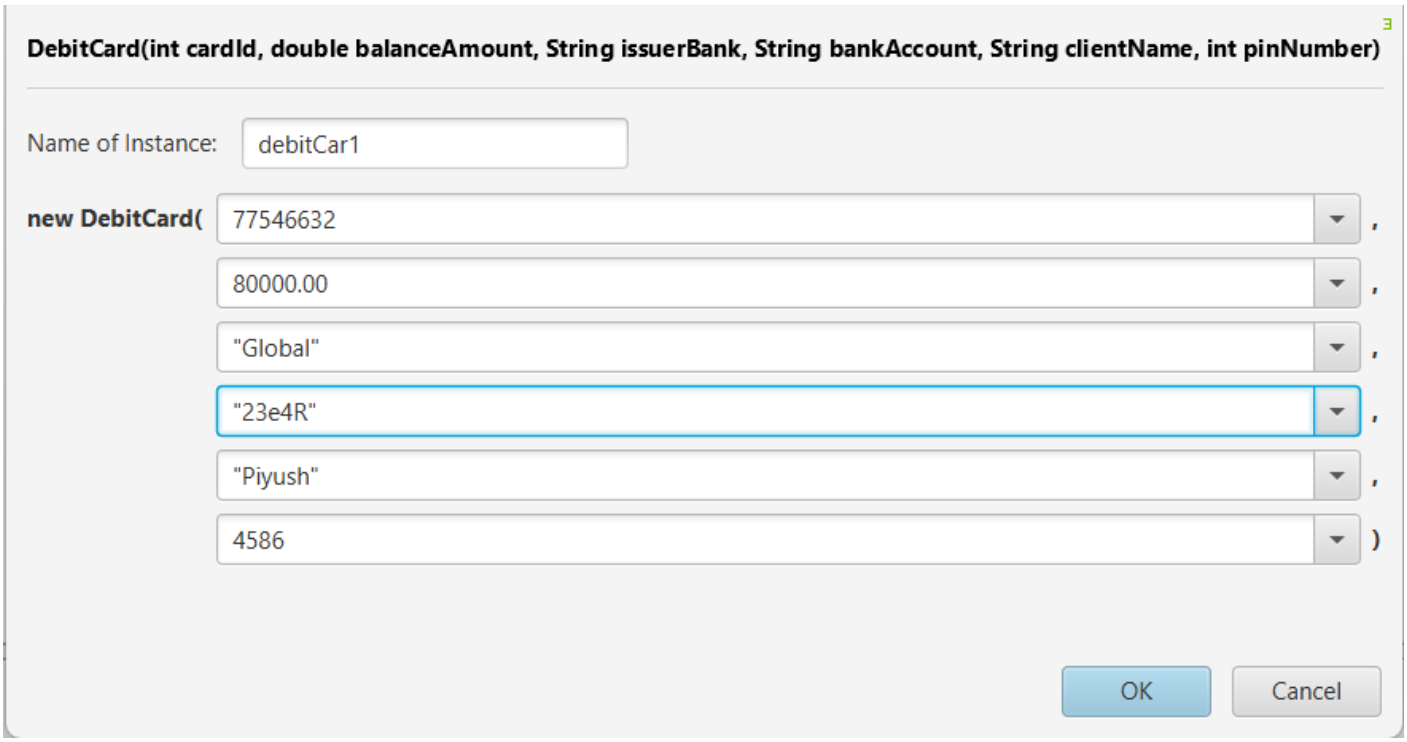

Figure 5 Detail Filling in Debit Class

Actual Output:

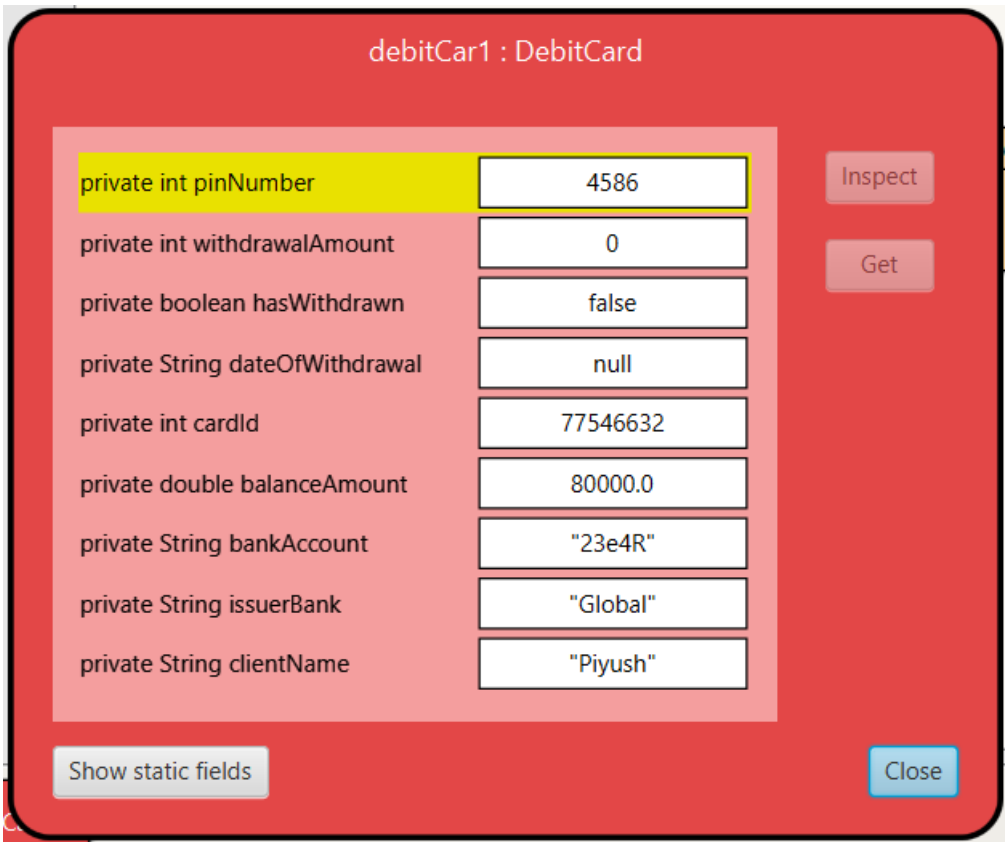

Figure 6 Initial Debit Card

Withdraw Method :

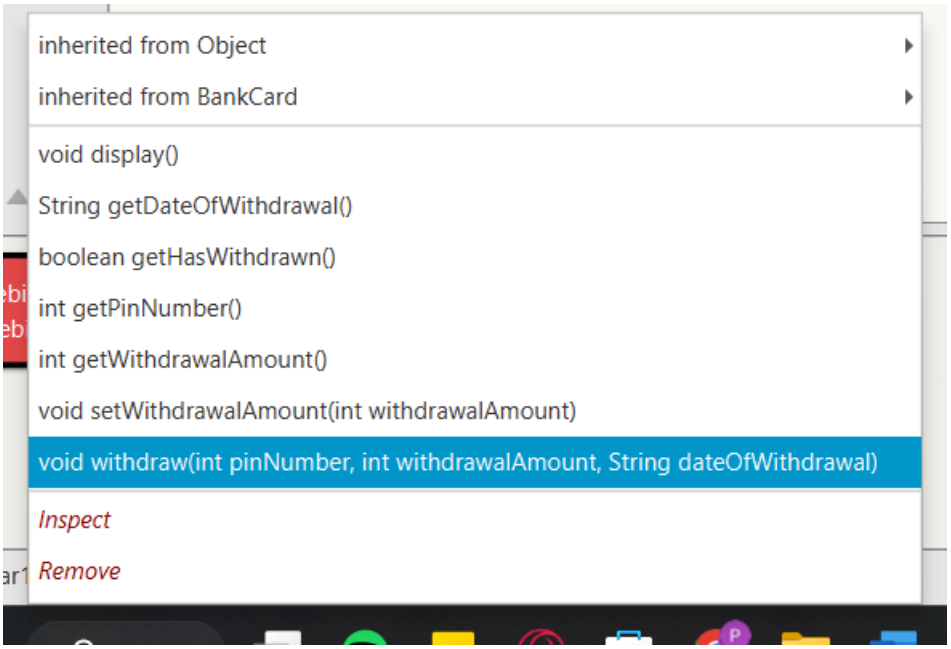

Figure 7 Using Withdraw Method

Expected Output:

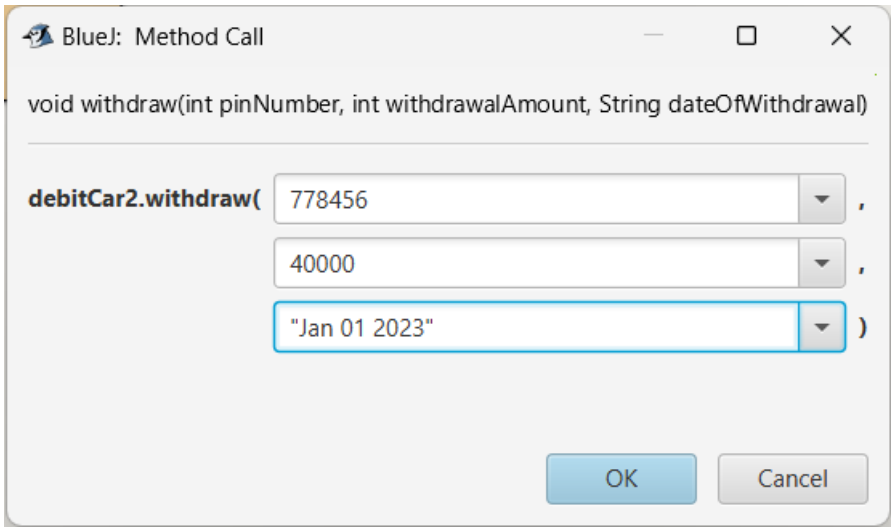

Figure 8 Filling Detail Of Withdraw

Actual Output:

Figure 9 Output After Filling Withdraw Method

since, pin doesn't match with actual pin: error message is shown.

Expected Output:

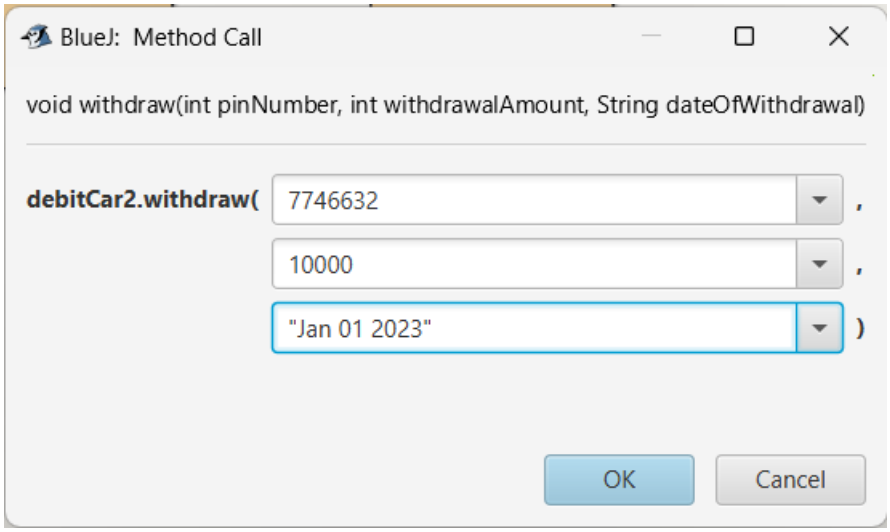

Figure 10 Filling Data in Withdraw

Actual Output:

Figure 11 Output After Filling Data

Re-inspecting the Debit Card Class:

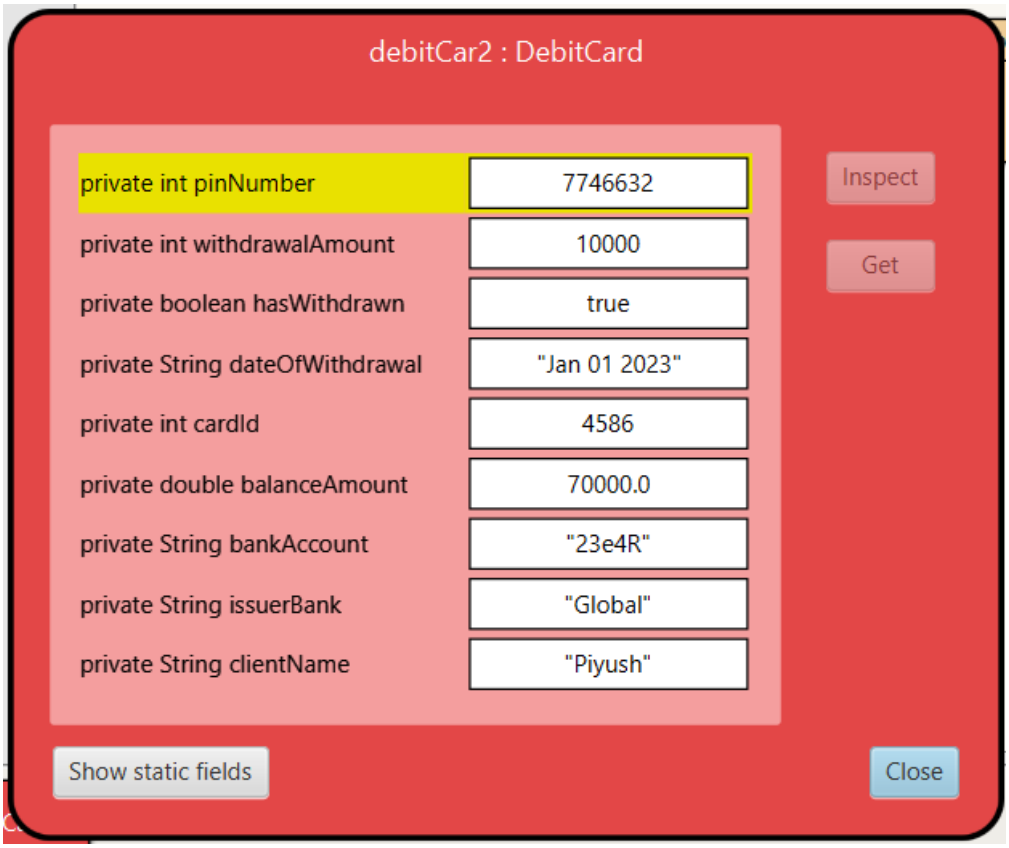

Figure 12 Re-Inspecting Debit Class

Table 1 Inspecting Debit Card Withdraw Method

Objective	**To check whether withdraw is successful**
Action	Checking with possible input user might enter
Expected Outcome	User should be able to withdraw
Actual Output	User has successfully withdrawn
Conclusion	Test successfully

5.2 Test 2:

In test 2, According to question we are supposed to Inspect Credit Card class, set the credit limit and reinspect the Credit Card Class.

Inspecting Credit Card Class:

Expected Output:

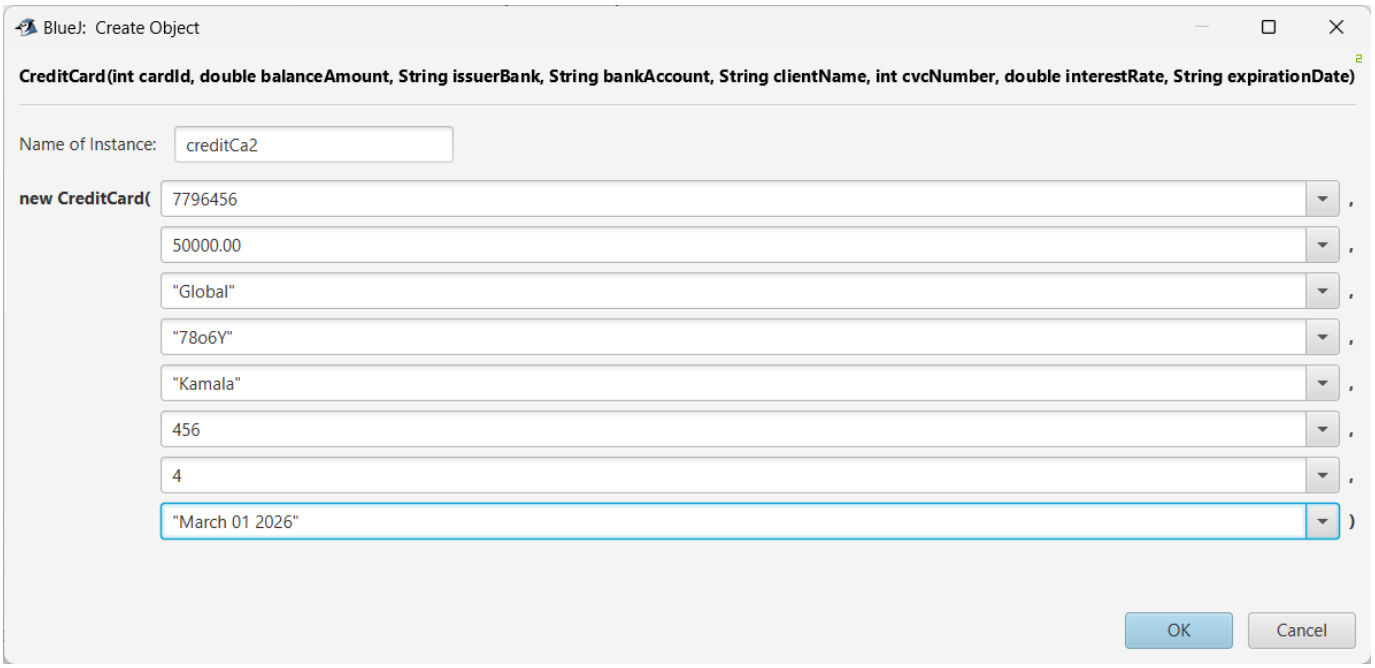

Figure 13 Filling Detail in Credit Class

Actual Output:

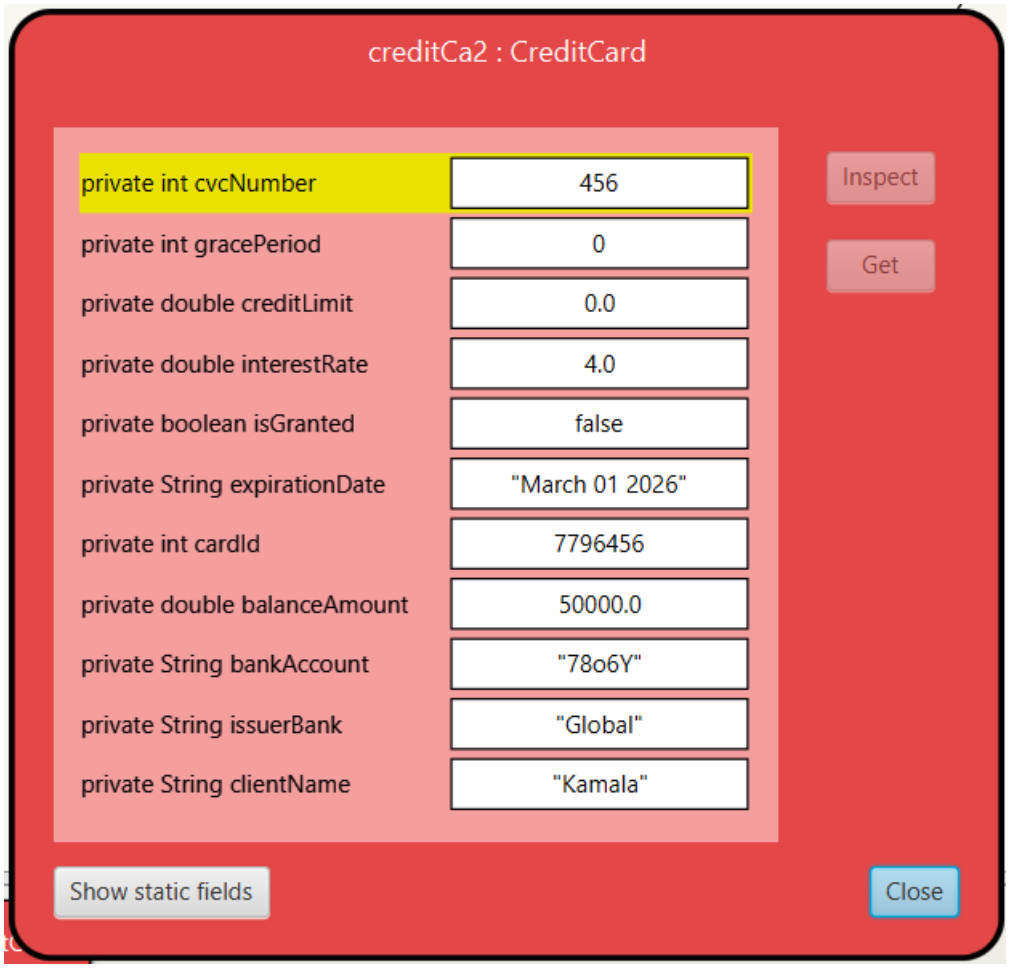

Figure 14 Inspecting Initial Credit Class

Setting CreditLimit Method:

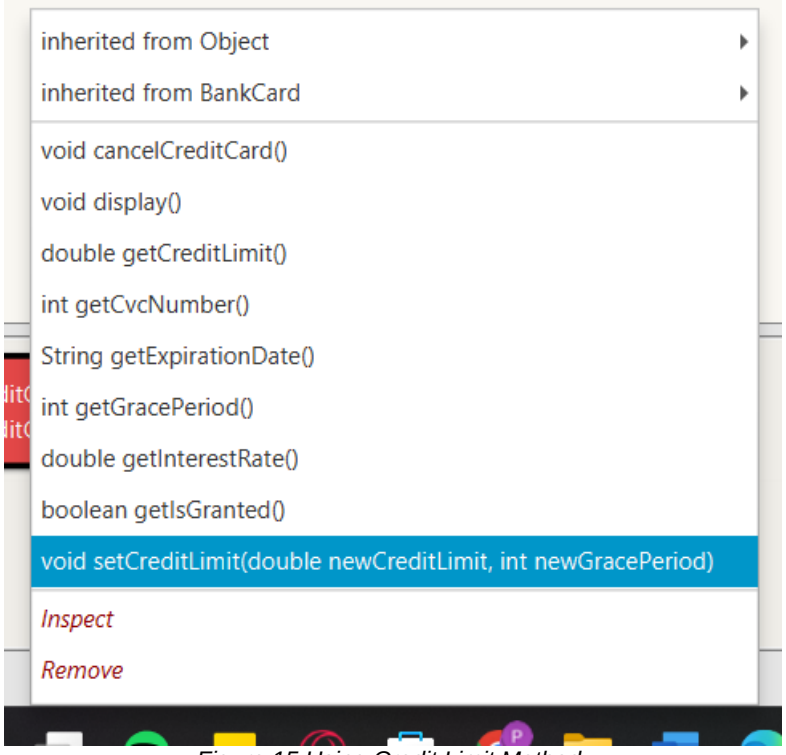

Figure 15 Using Credit Limit Method

Expected Output:

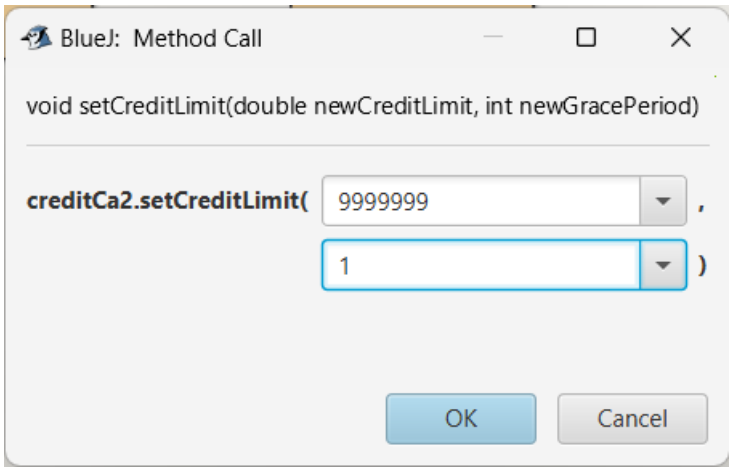

Figure 16 Filling Data in Credit Limit

Actual Output:

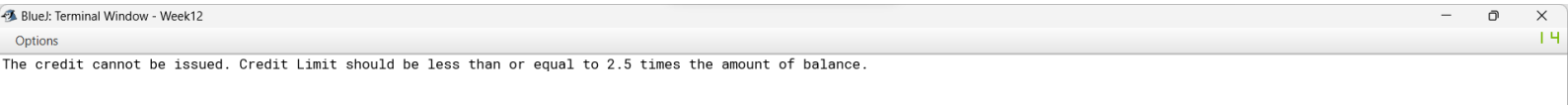

Figure 17 Output After Filling Data

Since the credit limit is more than 2.5 * Balance Amount, we are shown the suitable message.

Expected Output:

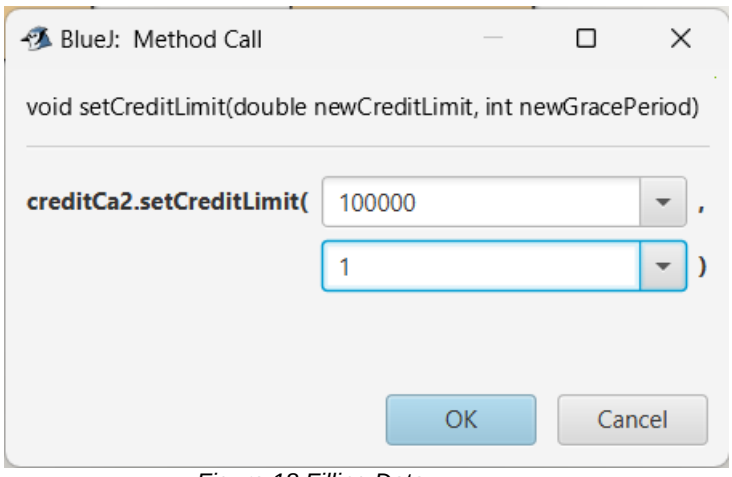

Figure 18 Filling Data

Final Output:

Re-inspecting Credit Class:

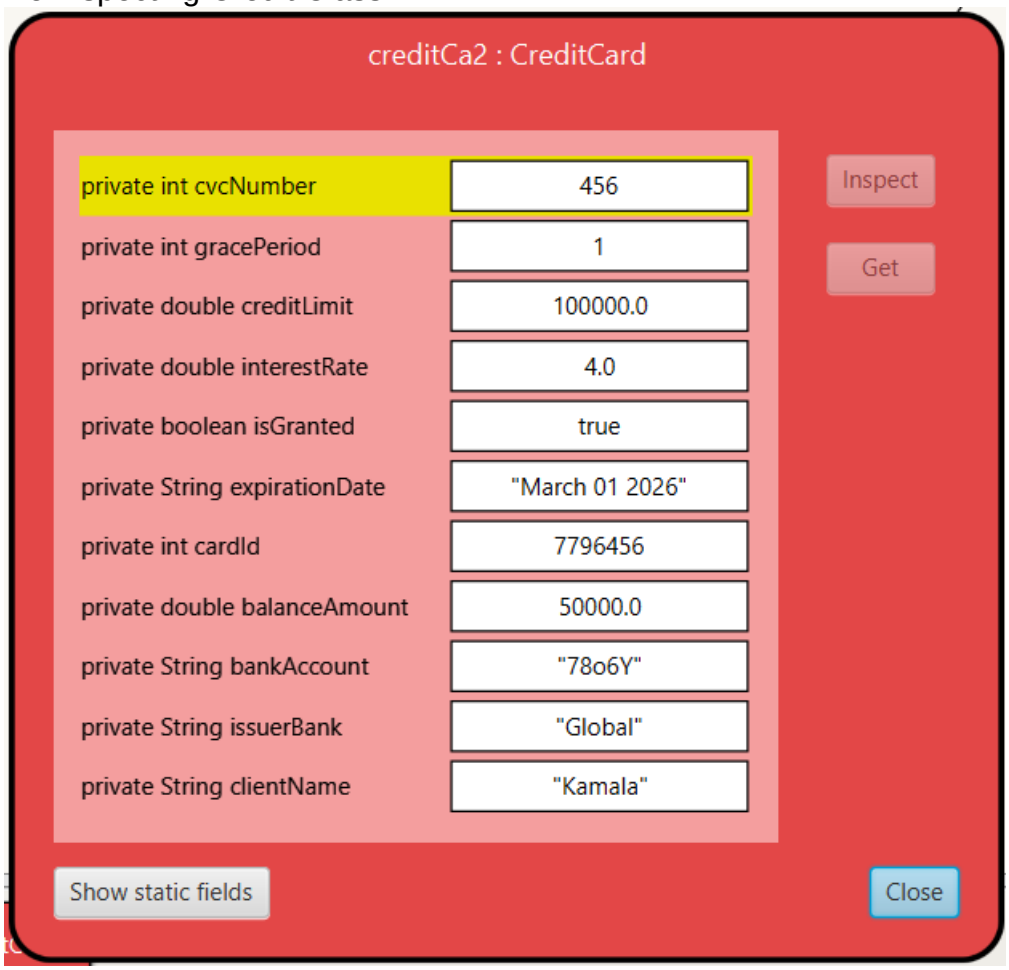

Figure 19 Re-Inspecting Credit Class

Table 2 Inspecting Credit Card, Credit Limit Method

Objective	**To Set Credit Limit**
Action	Inputting every possible input
Expected Outcome	User is successfully able to set credit limit
Actual Output	User has successfully set credit limit
Conclusion	Test successful

5.3 Test 3:

In test 3, according to question we are supposed to inspect credit class again after cancelling credit card.

Cancel Credit Card Method:

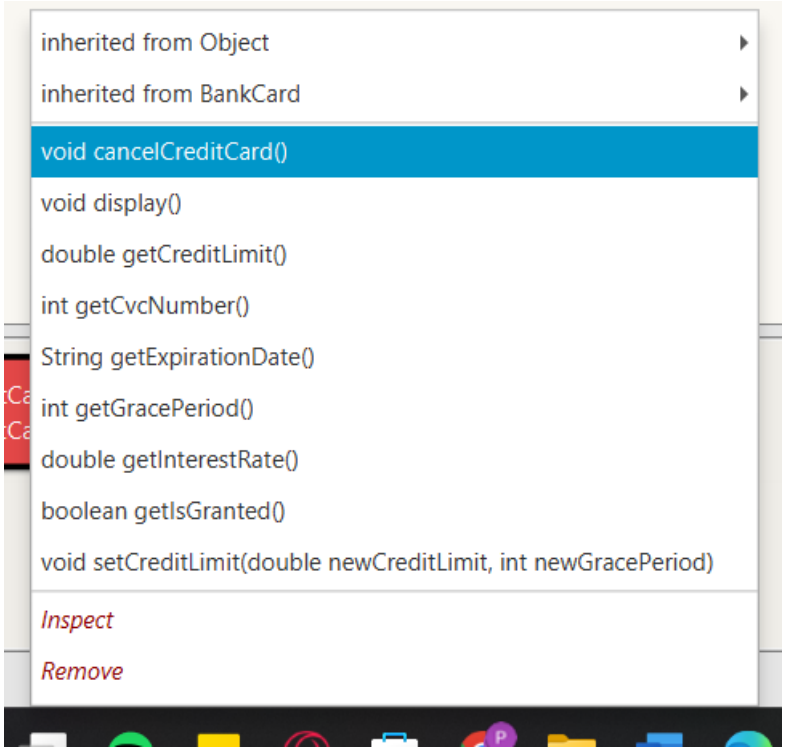

Figure 20 Using Cancel Credit Card Method

Re-inspecting Credit Class:

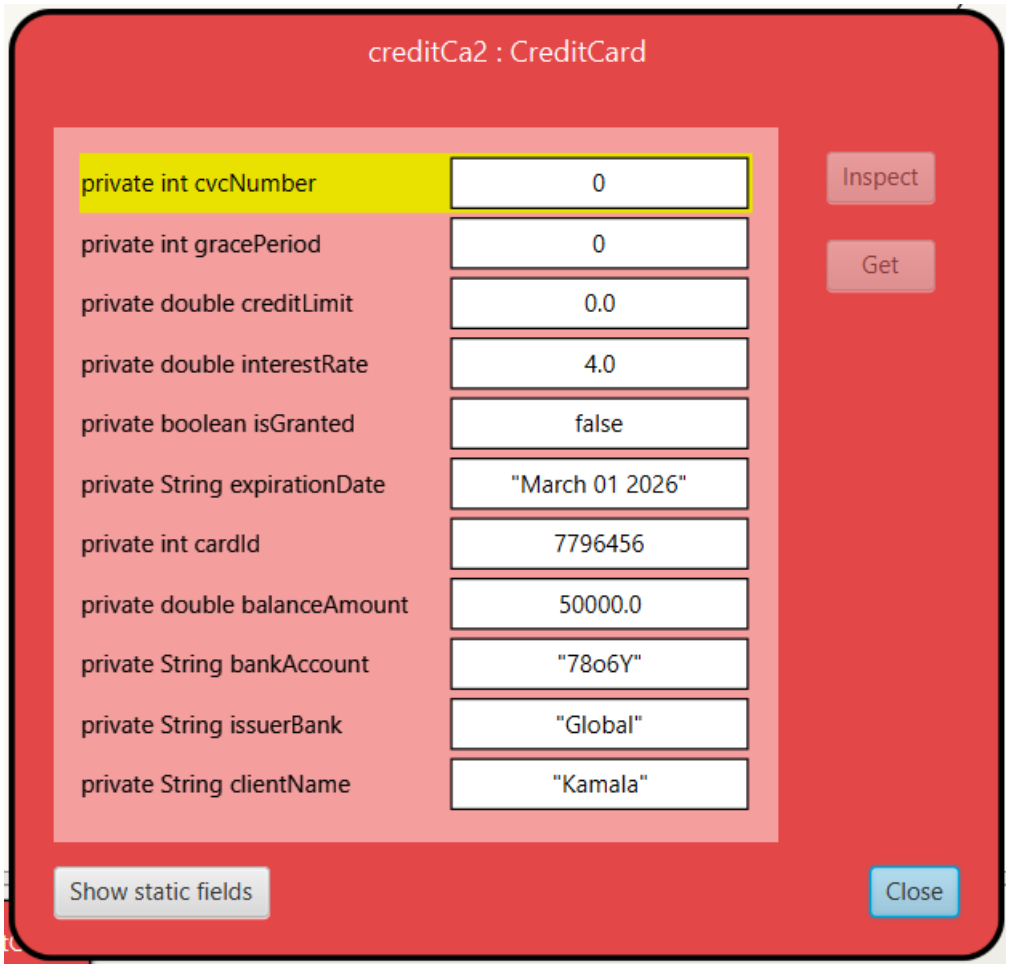

Figure 21 Re-Inspecting Credit Class

Table 3 Inspecting Credit Card Cancel Method

Objective	**To Cancel Credit Limit**
Action	Inputting every possible outcome
Expected Outcome	User should be able to cancel credit card
Actual Output	User has successfully cancel credit card
Conclusion	Test Successful

5.4 Test 4:

In test 4, We will be displaying all the details that we have gathered in previous test.

Debit Card Detail:

```
BlueJ: Terminal Window - Week12
Options
The client name is Piyush
The card ID is 4586
The balanceAmount is 80000.0
The issuer bank is Global
The bank account is 23e4R
Pin Number : 7746632
Amount of withdrawal : 0
Date Of Withdrawal: null
```

Figure 22 Initial Debit Card Display

Debit Card Display After Withdraw:

```
BlueJ: Terminal Window - Week12
Options
The client name is Piyush
The card ID is 4586
The balanceAmount is 70000.0
The issuer bank is Global
The bank account is 23e4R
Pin Number : 7746632
Amount of withdrawal : 10000
Date Of Withdrawal: Jan 01 2023
```

Figure 23 Debit Card Display after Withdraw

Credit Card Display:

```
BlueJ: Terminal Window - Week12
Options
The client name is Kamala
The card ID is 7796456
The balanceAmount is 50000.0
The issuer bank is Global
The bank account is 78o6Y
CVC Number: 456
Interest Rate: 4.0
Expiration Date: March 01 2026
```

Figure 24 Initial Credit Card Display

Credit Card Display After Credit Limit is set:

```
BlueJ: Terminal Window - Week12
Options
The client name is Kamala
The card ID is 7796456
The balanceAmount is 50000.0
The issuer bank is Global
The bank account is 78o6Y
CVC Number: 456
Credit Limit: 100000.0
Interest Rate: 4.0
Expiration Date: March 01 2026
Grace Period: 1
```

Figure 25 Credit Card Display after Setting Credit Limit

Credit Card Display After Cancel Credit Card:

```
BlueJ: Terminal Window - Week12
Options
The client name is Kamala
The card ID is 7796456
The balanceAmount is 50000.0
The issuer bank is Global
The bank account is 78o6Y
CVC Number: 0
Interest Rate: 4.0
Expiration Date: March 01 2026
```

Figure 26 Credit Card Display After Cancel Credit Card

Table 4 Display Method

Objective	**To Show All Display From Test 1,2,3**
Action	Use display() method to show outcome
Expected Outcome	User is able to see all outcome
Actual Output	Display was successfully shown to user
Conclusion	Test successfully

6. Error

In java error are defined in three factors; Compile Time Error, Runtime Error, and Logical Error. These errors help us to determine, how to solve the problem faster. During the project it was inevitable to not run into errors but with a bit of work I was able to debug the code and run it successfully. Here are some of the errors that I have ran into and solved during the projects.

6.1 Syntax Error

Errors where the compiler finds something wrong in your program, which makes it unable to execute the program. Like incorrect punctuation, or undeclared variable.

```
protected void setCreditLimit(double newCreditLimit, int newGracePeriod)
{
    if (newCreditLimit <= (2.5 * super.getBalanceAmount()) {
        creditLimit = newCreditLimit;
        gracePeriod = newGracePeriod;
        isGranted = true;
    }
    else {
    System.out.println("The credit cannot be issued. Credit Limit should be less than or equal to 2.5 times the amount of balance.");
    }
}
```

Figure 27 Syntax Error Problem Found

```
protected void setCreditLimit(double newCreditLimit, int newGracePeriod)
{
    if (newCreditLimit <= (2.5 * super.getBalanceAmount())) {
        creditLimit = newCreditLimit;
        gracePeriod = newGracePeriod;
        isGranted = true;
    }
    else {
    System.out.println("The credit cannot be issued. Credit Limit should be less than or equal to 2.5 times the amount of balance.");
    }
}
```

Figure 28 Syntax Error Problem Solved

6.2 Semantic Error

Semantic Error happens when we make error due to improper use of variables. And that's exactly what happens during my work in Bank Card:

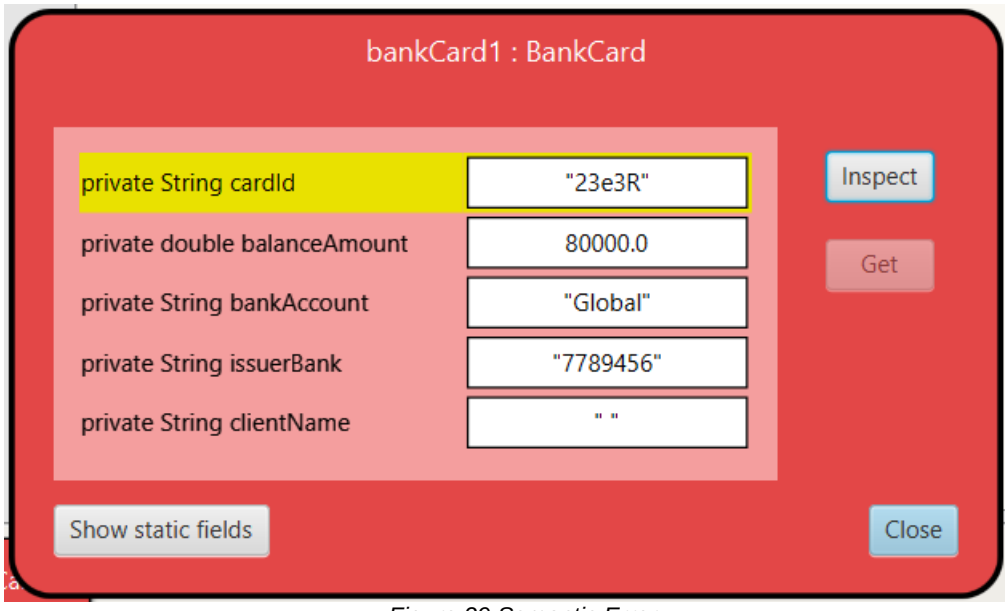

Figure 29 Semantic Error

As in the above diagram we can see a lot of error like bankAccount, issuerBank, and cardId are jumpled against each other this happens due to poor use of data type and variable.

```
private String cardId;
private double balanceAmount;
private String bankAccount;
private String issuerBank;
private String clientName;

//no args constructor
protected BankCard()
{

}

//this is parameterized contructor with four parameters
protected BankCard (String cardId, double balanceAmount, String issuerBank, String bankAccount)
{
    //initalizing variables
    this.cardId = cardId;
    this.balanceAmount = balanceAmount;
    this.bankAccount = bankAccount;
    this.issuerBank = issuerBank;
    this.clientName=" ";
}
```

Figure 30 Error Found

```java
public class BankCard
{
    //declaring variables
    private int cardId;
    private double balanceAmount;
    private String bankAccount;
    private String issuerBank;
    private String clientName;

    //no args constructor
    protected BankCard()
    {

    }

    //this is parameterized contructor with four parameters
    protected BankCard (int cardId, double balanceAmount, String issuerBank, String bankAccount)
    {
        //initalizing variables
        this.cardId = cardId;
        this.balanceAmount = balanceAmount;
        this.bankAccount = bankAccount;
        this.issuerBank = issuerBank;
        this.clientName=" ";
    }
```

Figure 31 Error Solved

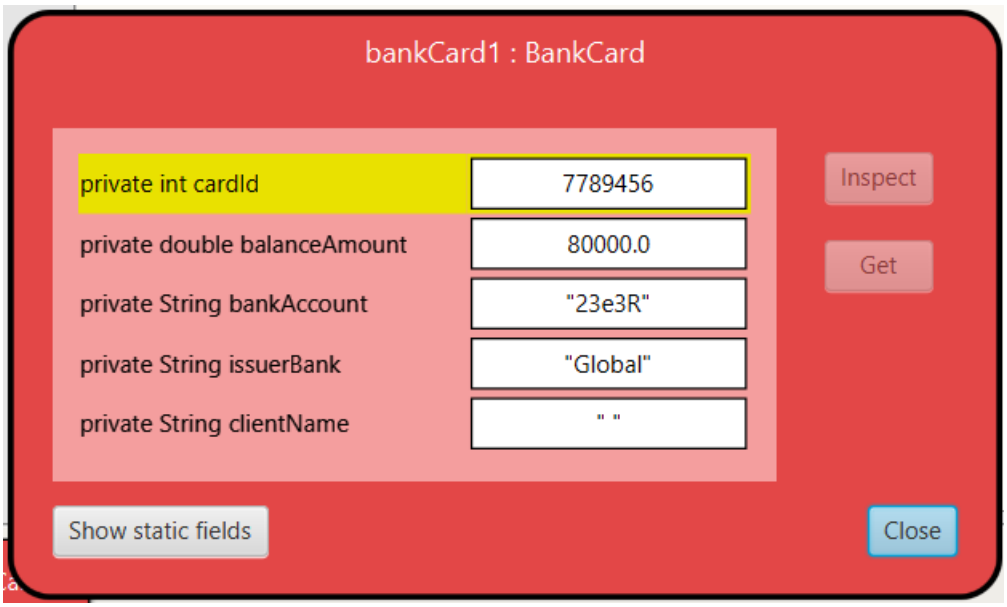

Figure 32 Error Solved

6.3 Logical Error

A logical error happens when program runs and compiles successfully but is unable to perform task accordingly.

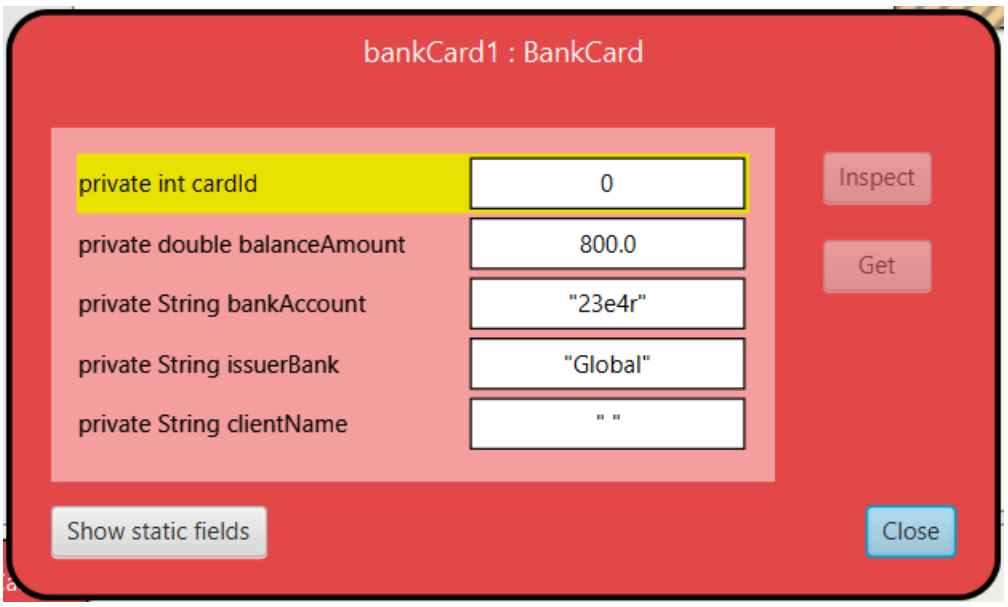

Figure 33 cardId not

```
protected BankCard(int cardID, double balanceAmount, String issuerBank, String bankAccount)
{
    this.cardId = cardId;
    this.balanceAmount = balanceAmount;
    this.bankAccount = bankAccount;
    this.issuerBank = issuerBank;
    this.clientName=" ";
}
```

Figure 34 Error Found

```
protected BankCard(int cardId, double balanceAmount, String issuerBank, String bankAccount)
{
    this.cardId = cardId;
    this.balanceAmount = balanceAmount;
    this.bankAccount = bankAccount;
    this.issuerBank = issuerBank;
    this.clientName=" ";
}
```

Figure 35 Error Solved

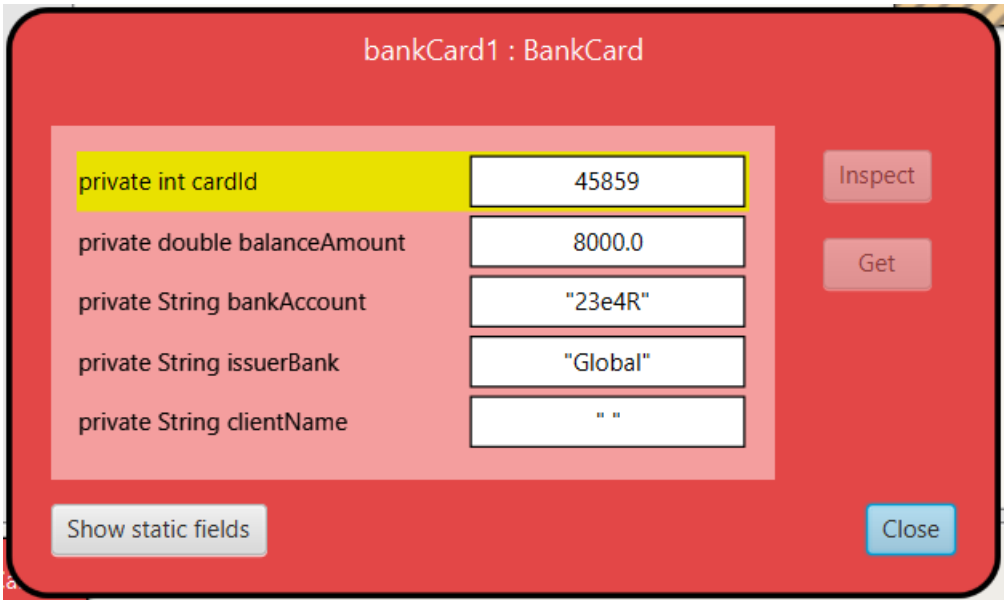

Figure 36 Checking if error re-occurs

Conclusion

By the end of the project, I was able to understand the concept of object-oriented concept a little better in programming whereas my report writing skill definitely improved. Breaking the part of programming and report writing skill, in programming I have gained more knowledge about inheritance, constructors and use of setter and getter better. While talking about report writing skill, I was able to be more on point and write important stuff in less words.

In this project, I had trouble in various parts even though I was familiar with concept. The most difficult part for me was finding logical errors that I had to debug and understanding in more basic about inheritance and constructors. Some of the problems that I had encountered I had asked help with our teachers, and friends. Resource that our teacher has given was really helpful as it contained most of the problem that a beginner like me would encounter.

This project was overall fun and most importantly informative which helped us on how to deal with real-world problems. And I will definitely will be able to tackle any question similar to this one.

Appendix:

Bank Card:

```
/**
 * This class is BankCard, super class of other two sub-class(DebitCard and CreditCard).
 *
 * @author (Prabal Gurung)
 * @version (5.1.0)
 */
public class BankCard
{
    //declaring variables
    private int cardId;
    private double balanceAmount;
    private String bankAccount;
    private String issuerBank;
    private String clientName;

    //no args constructor
    protected BankCard()
    {

    }

    //this is parameterized contructor with four parameters
    protected BankCard (int cardId, double balanceAmount, String issuerBank, String bankAccount)
    {
        //initalizing variables
        this.cardId = cardId;
        this.balanceAmount = balanceAmount;
        this.bankAccount = bankAccount;
        this.issuerBank = issuerBank;
        this.clientName=" ";
    }

    //getter
    protected int getCardId()
    {
        return cardId;
    }

    //getter
protected double getBalanceAmount()
    {
        return balanceAmount;
    }
```

```
    //getter
    protected String getBankAccount()
    {
        return bankAccount;
    }

    //getter
    protected String getIssuerBank()
    {
        return issuerBank;
    }

    //getter
    protected String getClientName()
    {
        return clientName;
    }

    //setter
    protected void setClientName (String newClientName)
    {
        this.clientName = newClientName;
    }

    //setter
    protected void setBalance (double newBalanceAmount)
    {
        this.balanceAmount = newBalanceAmount;
    }

    //this method function to print
    protected void display()
    {
        //checks whether client has filled name or not
        if(clientName !=  " ") {
            //prints following if "if" statement is true
            System.out.println("The client name is " + clientName);
            System.out.println("The card ID is " + cardId);
            System.out.println("The balanceAmount is " + balanceAmount);
            System.out.println("The issuer bank is " + issuerBank);
            System.out.println("The bank account is " + bankAccount);
        }
        else{
// prints suitable message if "if" statement is false
            System.out.println("Client name not assigned! Please enter the name and Try Again.");
        }
    }
}
```

Debit Card:

```
/**
 * This is sub-class of BankCard; DebitCard
 *
 * @author (Prabal Gurung)
 * @version (5.1.0)
 */
 public class DebitCard extends BankCard
 {
    //declaring variables
    private int pinNumber;
    private int withdrawalAmount;
    private boolean hasWithdrawn;
    private String dateOfWithdrawal;

    /*
     * this is parameterized constructor with six parameters
     * this method calls (cardId, balanceAmount , issuerBank , bankAccount) from its parent class
     * this method also initializes pinNumber
     * this method sets has withdrawn to false
     */
    protected DebitCard (int cardId, double balanceAmount , String issuerBank, String bankAccount,
String clientName, int pinNumber)
    {
        super(cardId, balanceAmount ,issuerBank ,bankAccount);
        super.setClientName(clientName);
        this.pinNumber = pinNumber;
        this.hasWithdrawn = false;
    }

    //getter
    protected int getPinNumber()
    {
        return pinNumber;
    }

    //getter
    protected int getWithdrawalAmount()
    {
        return withdrawalAmount;
    }

    //getter
    protected String getDateOfWithdrawal()
    {


   return dateOfWithdrawal;

   }
```

```
    //getter
    protected boolean getHasWithdrawn()
    {
        return hasWithdrawn;
    }

    //setter
    protected void setWithdrawalAmount(int withdrawalAmount)
    {
        this.withdrawalAmount = withdrawalAmount;
    }

    //this method function to withdraw
    protected void withdraw(int pinNumber, int withdrawalAmount, String dateOfWithdrawal)
    {
        //checks whether the entered pin is same as initial pin
        if (pinNumber == this.pinNumber)
        {
            //checks if balance amount is higher than withdraw amount
            if(super.getBalanceAmount() >= withdrawalAmount)
            {
                //performs following task if both statement is true
                hasWithdrawn = true;
                this.dateOfWithdrawal = dateOfWithdrawal;
                setBalance(getBalanceAmount() - withdrawalAmount);
                this.withdrawalAmount = withdrawalAmount;
                System.out.println("Withdrawal successful. New balance: " +
super.getBalanceAmount());
            }else{
                System.out.println("Insufficient fund");//prints suitable message if balance amount is
lower than withdrawal amount
            }
        }else{
            System.out.println("Invalid PIN");//prints suitable message if pin doesn't match to initail pin
        }
    }

    //this method function to print
    protected void display()
    {
        super.display();//prints attribute from parent class
        //checks whether hasWithdrawn is true or false
        if (hasWithdrawn = true ) {
            //prints the following if the condition is true
            System.out.println("Pin Number : "+ pinNumber);


     System.out.println("Amount of withdrawal : " + withdrawalAmount);

   System.out.println("Date Of Withdrawal: " + dateOfWithdrawal);
```

```
  }else{
        System.out.println("Balance Amount: " + super.getBalanceAmount());//prints balance amount from super class if condiotion is false
      }
   }
}
```

Credit Card:

```
/**
 * Credit Card is sub-class of Bank Card and has detail of credit card according to question
 *
 * @author (Prabal Gurung)
 * @version (5.1.0)
 */
public class CreditCard extends BankCard
{
    //declaring varaibles
    private int cvcNumber;
    private int gracePeriod;
    private double creditLimit;
    private double interestRate;
    private boolean isGranted;
    private String expirationDate;

    /*
     * this is parameterized constructor with eight parameters
     * this method calls (cardId, balanceAmount ,issuerBank ,bankAccount, client name) from its parent class
     * this method also initializes (cvc number, interest rate, expiration date)
     * this method sets is granted to false
     */
    protected CreditCard (int cardId , double balanceAmount, String issuerBank, String bankAccount, String clientName, int cvcNumber, double interestRate, String expirationDate)
    {
        super(cardId, balanceAmount, issuerBank, bankAccount);
        super.setClientName(clientName);
        this.cvcNumber = cvcNumber;
        this.interestRate = interestRate;
        this.expirationDate = expirationDate;
        this.isGranted = false;
    }

    //getter
    protected int getCvcNumber()
    {
        return cvcNumber;
    }

    //getter
    protected double getCreditLimit()
    {
        return creditLimit;
    }
```

```
//getter
  protected double getInterestRate()
  {
     return interestRate;
  }

  //getter
  protected String getExpirationDate()
  {
     return expirationDate;
  }

  //getter
  protected int getGracePeriod()
  {
     return gracePeriod;
  }

  //getter
  protected boolean getIsGranted()
  {
     return isGranted;
  }

  //setter
  protected void setCreditLimit(double newCreditLimit, int newGracePeriod)
  {
     //checks if entered credit limit is under maximum
     if (newCreditLimit <= (2.5 * super.getBalanceAmount())) {
        //performs following function if true
        creditLimit = newCreditLimit;
        gracePeriod = newGracePeriod;
        isGranted = true;
     }
     else {
     System.out.println("The credit cannot be issued. Credit Limit should be less than or equal to
2.5 times the amount of balance.");//if false following message is displayed
     }
  }

  //voids creditcard
  protected void cancelCreditCard() {
     cvcNumber = 0; //intializes cvcNumber to zero
     creditLimit = 0; // initializes creditLimit to zero
     gracePeriod = 0; // initializes gracePeriod to zero
     isGranted = false; // set isGranted to false
  }
```

```
 //this method function to print

public void display() {
        super.display(); //prints display method from super class
        System.out.println("CVC Number: " + cvcNumber); //prints cvcNumber
        System.out.println("Interest Rate: " + interestRate); //prints interest rate
        System.out.println("Expiration Date: " + expirationDate); //prints expiration date
        //checks if isGranted value is true
        if (isGranted == true) {
            //if isGranted is set to true then prints following additionally
            System.out.println("Credit Limit: " + creditLimit);
            System.out.println("Grace Period: " + gracePeriod);
        }
    }
}
```