CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Ayub Bhatta

London Met ID: 22067809

College ID: np01cp4a220193

Group: L1C8

Assignment Due Date: Friday, January 27, 2023

Assignment Submission Date: Thursday, January 26, 2023

Table Of Content

Table Of Figure

1.Introduction:

1.1 Introduction Project Content:

Java is a Object Oriented Programming(OOP) language which is fairly easy to use and learn.

This coursework is designed to get us, college students, to know how to use and implement Java programming, specifically OOP(Object Oriented Programming). It is designed the way to help us get used to this programming language and to get used to using Blue J, a loose Java Development Environment designed to beginners, utilised by many worldwide.

This coursework asks us to create a class named BankCard, which has five attributes, which correspond to the card Id, client name, issuer bank, bank account, and BalanceAmount. The client name, issuer bank, bank account are each represented as a string of text and Card ID, and balance amount as a number.

This class named BankCard have two child classes named DebitCard and CreditCard, where Debitcard has Pin number, withdrawal amount, date of withdrawal and has withdrawn as attributes and CreditCard has Cvc number, credit limit, interest rate, expiration date, grace period and isGranted as attributes. These attributes will be stored in respective method and will be displayed from a method named display.

1.2 Tools Used:

1.2.1 Blue J:

The BlueJ terrain was developed as part of a university exploration design about tutoring object-exposure to newcomers. The end of blueJ is to give an easy-to -use tutoring terrain for the java language that facilitates the tutoring of java of first-time scholars. Special emphasis has been placed on visualisation and commerce ways to produce a largely interactive terrain that encourage trial and disquisition.

It was developed to help people to learn Object Oriented Program (OOP). The objects can be tested. BlueJ has a simpler user interface than other professional IDEs. It has a variety of tools that are tailored to its objectives. Standard development tools are available such as an editor, compiler and run time etc.

1.2.2 MS-Word:

Microsoft word is a word processor application software which was developed by Microsoft in 1983. Ms-Word has advanced features which allow us to edit and format our files and documents in the best possible way. Ms-word enables user to do write document, create documents, resumes, contracts etc. This is commonly used programs in both office and home. It was developed to help people to make projects.

2. Class Diagram:

.2.1 BankCard:

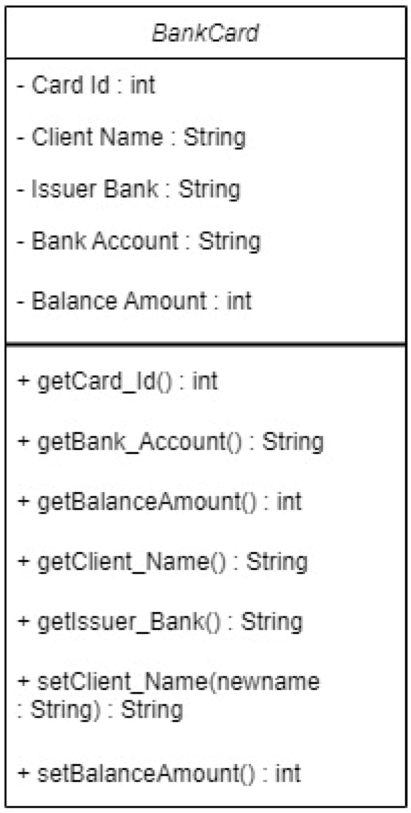

BankCard
- Card Id : int
- Client Name : String
- Issuer Bank : String
- Bank Account : String
- Balance Amount : int
+ getCard_Id() : int
+ getBank_Account() : String
+ getBalanceAmount() : int
+ getClient_Name() : String
+ getIssuer_Bank() : String
+ setClient_Name(newname : String) : String
+ setBalanceAmount() : int

Figure 1: BankCard

2.2 DebitCard:

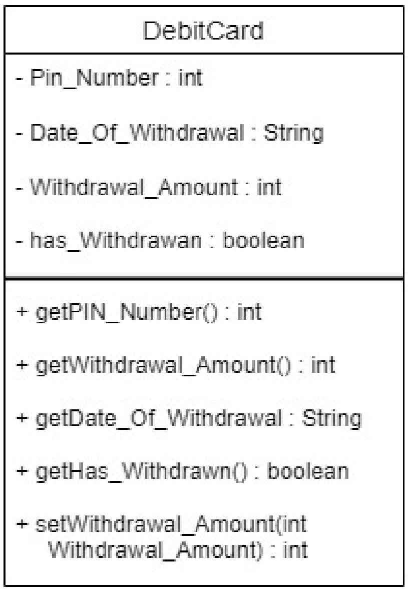

DebitCard
- Pin_Number : int
- Date_Of_Withdrawal : String
- Withdrawal_Amount : int
- has_Withdrawan : boolean
+ getPIN_Number() : int
+ getWithdrawal_Amount() : int
+ getDate_Of_Withdrawal : String
+ getHas_Withdrawn() : boolean
+ setWithdrawal_Amount(int Withdrawal_Amount) : int

Figure 2: DebitCard

2.3 CreditCard:

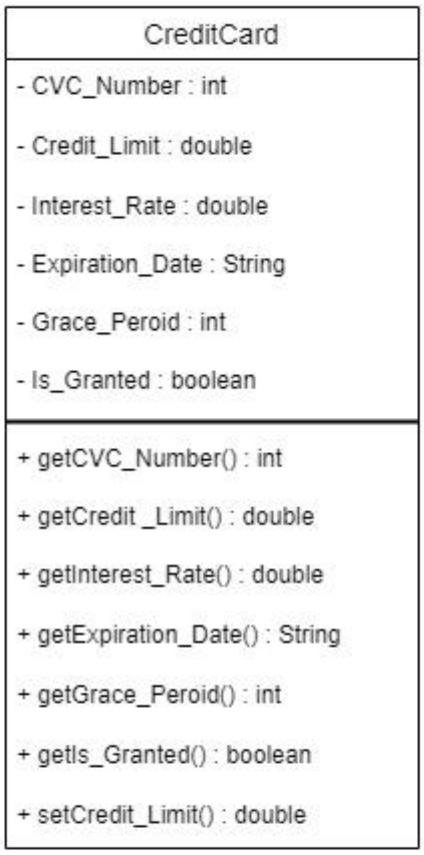

CreditCard
- CVC_Number : int
- Credit_Limit : double
- Interest_Rate : double
- Expiration_Date : String
- Grace_Peroid : int
- Is_Granted : boolean
+ getCVC_Number() : int
+ getCredit _Limit() : double
+ getInterest_Rate() : double
+ getExpiration_Date() : String
+ getGrace_Peroid() : int
+ getIs_Granted() : boolean
+ setCredit_Limit() : double

Figure 3:CreditCard

2.4 Partial View Of Class Diagram:

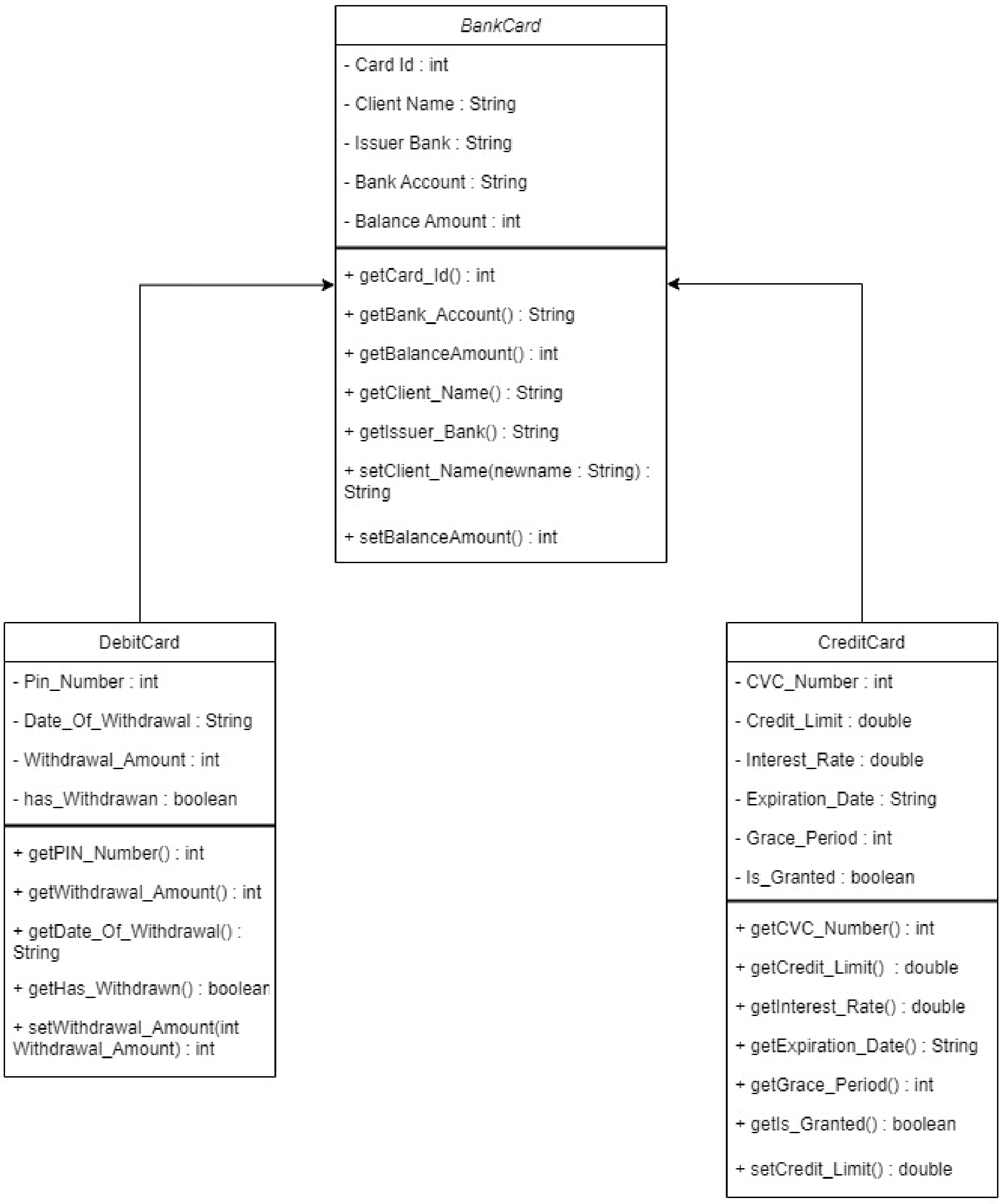

BankCard
- Card Id : int
- Client Name : String
- Issuer Bank : String
- Bank Account : String
- Balance Amount : int
+ getCard_Id() : int
+ getBank_Account() : String
+ getBalanceAmount() : int
+ getClient_Name() : String
+ getIssuer_Bank() : String
+ setClient_Name(newname : String) : String
+ setBalanceAmount() : int
DebitCard
- Pin_Number : int
- Date_Of_Withdrawal : String
- Withdrawal_Amount : int
- has_Withdrawan : boolean
+ getPIN_Number() : int
+ getWithdrawal_Amount() : int
+ getDate_Of_Withdrawal() : String
+ getHas_Withdrawn() : boolean
+ setWithdrawal_Amount(int Withdrawal_Amount) : int
CreditCard
- CVC_Number : int
- Credit_Limit : double
- Interest_Rate : double
- Expiration_Date : String
- Grace_Period : int
- Is_Granted : boolean
+ getCVC_Number() : int
+ getCredit_Limit() : double
+ getInterest_Rate() : double
+ getExpiration_Date() : String
+ getGrace_Period() : int
+ getIs_Granted() : boolean
+ setCredit_Limit() : double

Figure 4:Partial view of class diagram

3. Pseudocode:

3.1 BankCard :

```
Start
        CREATE Parent class BankCard
DO
        DECLARE instance variable Card_Id as a data type int as private
        DECLARE instance variable Client_Name as a data type String as private
        DECLARE instance variable BalanceAmount as a data type int as private
        DECLARE instance variable Client_Name as a data type String as private
        DECLARE instance variable Issuer_Bank as a data type String as private
END DO

        CREATE a constructor of the BankCard class with parameters
        (BalanceAmount, Card_Id, bank_account, Issuer_Bank)

        UPDATE instance variable BalanceAmount with parameter
        BalanceAmount

        UPDATE instance variable Card_Id with parameter Card_Id

        UPDATE instance variable bank_account with parameter
        bank_amount

        UPDATE instance variable Issuer_Bank with parameter Issuer_Bank


        CREATE getter method for instance variable Card_Id with return type
        int
```

```
DO
        RETURN the value of instance variable Card_Id
END DO
        CREATE getter method for instance variable Bank_Account with return
        type String
DO
        RETURN the value of instance variable Bank_Account
END DO
        CREATE getter method for instance variable BalanceAmount with
        return type int
DO
        RETURN the value of instance variable BalanceAmount
END DO
        CREATE getter method for instance variable Client_Name with return
        type String
DO
        RETURN the value of instance variable Client_Name
END DO
        CREATE getter method for instance variable Issuer_Bank with return
        type String
DO
        RETURN the value of instance variable Issuer_Bank
END DO
        CREATE setter method for Client_Name with parameter Client_Name
        of data type String
DO
```

```
        SET instance variable Client_Name with parameter value of Client_Name

    END DO

    CREATE a display method
    DO
        IF Client_Name is an empty string
            PRINT “Error”
        END IF
    END DO
        ELSE
    DO
        PRINT “Card Id: ” with the value of the instance variable Card_Id
        PRINT “Bank Amount: ” with the value of the instance variable Bank_Amount
        PRINT “Balance Amount: ” with the value of the instance variable BalanceAmount
        PRINT “Client Name: ” with the value of the instance variable Client_Name
        PRINT “Issuer Bank:  ” with the value of the instance variable Issuer_Bank
    END DO
END
```

3.2 DebitCard:

```
START
    CREATE first child class DebitCard which extends BankCard
DO
    DECLARE instance variable PIN_Number as data type int
    DECLARE instance variable Withdrawal_Amount as data type int
    DECLARE instance variable Date_Of_Withdrawal as data type String
    DECLARE instance variable Has_Withdrawan as data type Boolean
 END DO
    CREATE a constructor of the DebitCard class with parameters
    (BalanceAmount, Card_Id, bank_account, Issuer_Bank, Client_Name,
    PIN_Number)
    CALL a super of (BalanceAmount, Card_Id, bank_account,
    Issuer_Bank)
    SET the value of the Client_Name through parameter Client_Name
    UPDATE instance variable PIN_Number equals to PIN_Number using
    THIS keyword
    UPDATE instance variable Has_Withdrawan as False using THIS
    keyword
    CREATE getter method for instance variable PIN_Number with return
    type int
DO
    RETURN the value of instance variable PIN_Number
END DO
```

```
    CREATE getter method for instance variable Withdrawal_Amount with
    return type int
DO
    RETURN the value of instance variable Withdrawal_Amount
END DO
    CREATE getter method for instance variable Date_Of_Withdrawal with
    return type String
DO
    RETURN the value of instance variable Date_Of_Withdrawal
END DO
    CREATE getter method for instance variable Has_Withdrawn with
    return type boolean
DO
    RETURN the value of instance variable Has_Withdrawn
END DO
    CREATE setter method for instance variable Withdrawal_Amount with
    parameter Withdrawal_Amount of data type int
DO
    SET instance variable Withdrawal_Amount with the parameter value of
    Withdrawal_Amount
END DO
    CREATE a method Withdraw
    IF PIN_Number is equals to PIN_Number
    DO
        IF Withdrawal_Amount is smaller than or equals to get
    BalanceAmount()
```

```
                THIS Withdrawal_Amount is equals to Withdrawal_Amount
                THIS Date_Of_Withdrawal is equals to Date_Of_Withdrawal
                THIS Has_Withdrawan is true
                ELSE
                    PRINT “Insufficient Balance!”
                END IF
            ELSE
                PRINT “Invalid Pin!”
            END IF
            END DO
                CREATE a display method
                CALL the display method of parent class
                PRINT “PIN_Number: “ with the value of instance variable
        PIN_Number

            DO
            IF
                Has_Withdrawan
                PRINT "Withdrawal Amount: " with the value of instance variable
        Withdrawal_Amount
                PRINT "Date of Withdrawal: " with the value of instance variable
        Date_Of_Withdrawal
            END IF
            ELSE
                PRINT “Pending Transaction."
```

```
END DO

END
```

3.3 CreditCard:

```
START
        CREATE second child class CreditCard which extends BankCard
DO
        DECLARE instance variable CVC_Number as data type int
        DECLARE instance variable Credit_Limit as data type double
        DECLARE instance variable Interest_Rate as data type double
        DECLARE instance variable Expiration_Date as data type String
        DECLARE instance variable Garce_Period as data type Integer
        DECLARE instance variable Is_Granted as data type Booolean
 END DO
        CREATE a constructor of the CreditCard class with parameters
        (Card_Id, Client_Name, Issuer_Bank, Bank_Account,
        Balance_Amount, CVC_Number,
        CALL a super of (Balance_Amount, Card_Id, Issuer_Bank,
        Bank_Account)
        THIS CVC_Number is equals to CVC_Number
        THIS Interest_Rate is equals to Interest_Rate
        THIS Expiration_Date is equals to Expiration_Date
        SET the value of Client_Name as Client_Name
        UPDATE instance variable Is_Granted as False
        CREATE getter method for instance variable CVC_Number with return
        type int
DO
```

```
        RETURN the value of instance variable CVC_Number
END DO
        CREATE getter method for instance variable Credit_Limit with return
        type Double
DO
        RETURN the value of instance variable Credit_Limit
END DO
        CREATE getter method for instance variable Interest_Rate with return
        type Double
DO
        RETURN the value of instance variable Interest_Rate
END DO
        CREATE getter method for instance variable Expiration_Date with
        return type String
DO
        RETURN the value of instance variable Expiration_Date
END DO
        CREATE getter method for instance variable Grace_Period with return
        type int
DO
        RETURN the value of instance variable Grace_Period
END DO
        CREATE getter method for instance variable Is_Granted with return
        type Boolean
DO
        RETURN the value of instance variable Is_Granted
```

```
END DO
DO
        SET instance variable Credit_Limit with the parameter value of
        Credit_Limit and Grace_Period
IF
        Credit_Limit  is smaller or equals to twice and a half of BalanceAmount
        CALL Credit_Limit is equals to Credit_Limit
        CALL Grace_Period is equals to Grace_Period
        UPDATE instance variable Is_Granted to true
END IF
ELSE
        PRINT "Credit cannot be issued. Please check your account and try
again."
END DO


        CREATE a public method cancelCreditCard
DO
IF      the value of instance variable Is_Granted is True
        UPDATE the value of instance variable THIS CVC_Number equals to
zero.
        UPDATE the value of instance variable THIS Credit_Limit equals to
zero.
        UPDATE the value of instance variable THIS Grace_Period equals to
zero.
        UPDATE instance variable THIS Is_Granted to false.
END IF
```

```
    ELSE
        PRINT "Credit card is not active."


        CREATE a display method
        CALL the display method of parent class
    DO
    IF
        Is_Granted
        PRINT "CVC Number: "    with the value of instance variable CVC_Number
        PRINT “CreditLimit:“ with the value of instance variable Withdrawal_Amount
        PRINT "InterestRate:   " with the value of instance variable Interest_Rate
        PRINT”ExpirationDate:"with the value of instance variable Expiration_Date
        PRINT "Grace Period: "with the value of instance variable Grace_Period
    END IF
    ELSE
        PRINT "Credit card is not granted."
    END DO
    END
```

4.Method Description:

To complete this Course Work, many methods were used in each class. Short description of the method used in this coursework are given below:

4.1BankCard class:

- **BankCard(int BalanceAmount,int Card_Id, String bank_account, String Issuer_Bank)**

  This method assigns the parameter of the BankCard to the respective new values.

- **getCard_Id()**

  This is an accessor method that will extract the instance variable Card_Id from private int and returns the value.

- **getBank_Account()**

  This is an accessor method that will extract instance variable Bank_Account from private String and returns the value.

- **getBalanceAmount()**

  This is an accessor method that will extract instance variable BalanceAmount from private String and returns the value.

- **getClient_Name()**

  This is an accessor method that will extract instance variable Client_Name from private String and returns the value.

- **getIssuer_Bank()**

  This is an accessor method that will extract instance variable Issuer_Bank from private String and returns the value.

- **setClient_name(String Client_Name)**

  This is a mutator method that takes Client_Name as a parameter and set the value of the client_Name to the Client_name.

- **Display():**

  This is a method that display the details of the BankCard which are Card_Id, Bank_Account, BalanceAmount, Client_Name and Issuer_Bank, whose values are taken from instance variables.

4.2 DebitCard Class:

- **DebitCard(int BalanceAmount, int Card_Id, String bank_account, String Issuer_Bank, String Client_Name, int PIN_Number)**

  This method assigns the parameters of the DebitCard to the respective new values.

- **getPIN_Number( )**

  This is an accessor method that will extract instance variable form PIN_Number private int and return its value.

- **getWithdrawal_Amount( )**

  This is an accessor method that will extract instance variable form Withdrawal_Amount private int and return its value.

- **getDate_Of_Withdrawal( )**

  This is an accessor method that will extract instance variable form Date_Of_Withdrawal private int and return its value.

- **getHas_Withdrawn( )**

  This is an accessor method that will extract instance variable form Has_Withdrawn private int and return its value.

- **setWithdrawal_Amount(int Withdrawal_Amount)**

  This is a mutator method that takes Withdrawal_Amount as a parameter and set the value of the Withdrawal_Amount to the Withdrawal_Amount.

- **Withdraw( )**
  This method will withdraw some amount of money if the pin number is correct, print "Insufficient balance!" if the withdrawal amount is higher than the balance amount and will print “Invalid Pin!” if the Pin number is wrong.

- **Display( )**

  This method will call the BankCard display method using **Super** to display the BalanceAmount, Card_Id, bank_account and Issuer_Bank along with Pin number, Withdrawal amount and Date of withdrawal if it has withdrawn or it will print “Pending Transaction”.

4.3 CreditCard Class :

- **CreditCard(int Card_Id, String Client_Name, String Issuer_Bank, String Bank_Account, int Balance_Amount, int CVC_Number, double Interest_Rate, String Expiration_Date)**

  This method assigns the parameters of the CreditCard to the respective new values.

- **getCVC_Number()**

  This is an accessor method that will extract instance variable from CVC_Number private int and returns value.

- **getCredit_Limit()**

  This is an accessor method that will extract instance variable from Credit_Limit private double and returns value.

- **getInterest_Rate**

  This is an accessor method that will extract instance variable from Interest_Rate private double and returns value.

- **getExpiration_Date()**

  This is an accessor method that will extract instance variable from Expiration_Date private String and returns value.

- **getGrace_Period()**

  This is an accessor method that will extract instance variable from Grace_Period private int and returns value.

- **getIs_Granted()**

  This is an accessor method that will extract instance variable from Is_Granted Period private boolean and returns value.

- **setCredit_Limit(double new_credit_limit, int new_grace_period)**

  This is a mutator method that takes Credit_Limit as a parameter and set the value of the Credit_Limit to the new_credit_limit and new_grace_period.

- **cancelCreditCard()**

  This method will cancel credit card if Is_Granted is true, it will print cvc number equals zero, credit limit equals zero, grace period equals zero and will alse print that the credit card is not granted else it will print Credit card is not active.

- **Display()**

  This method will call the vehicle display method using Super to displaythe value of CVC_Number, Credit_Limit, Interest_Rate, Expiration_Date and Grace_Period if the Credit_Card is granted else it will print that the CreditCard is not granted.

5. Testing

5.1 Table 1:

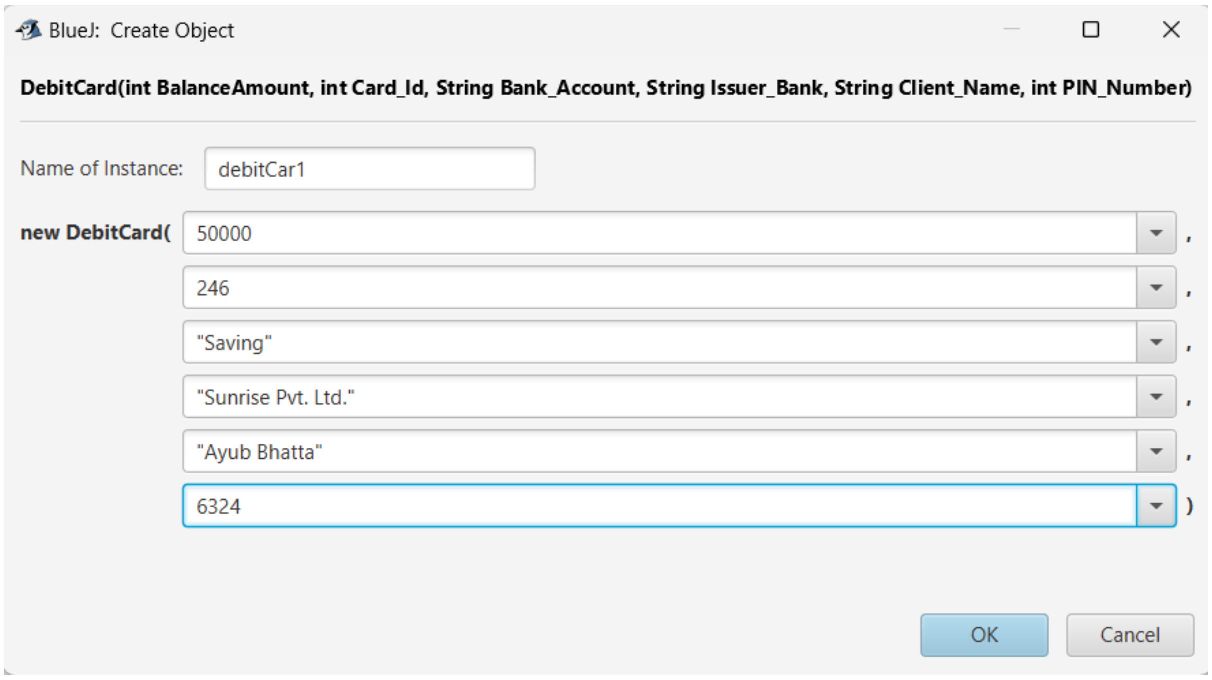
Inspect the Debit Card class, withdraw the amount, and re-inspect the Debit Card Class.

Test-1	
Objective	To inspect Debit Card class, withdraw the amount and re-inspect the Debit Card class

Action	Create object of Debit Card class with parameters:
	BalanceAmount = 50000
	Card_Id = 246
	Bank_Account = “Saving”
	Issuer_Bank = “Sunrise Pvt. Ltd.”
	Client_Name = “Ayub Bhatta”
	PIN_Number = 6324
	Call the withdraw amount with parameter
	WithdrawalAmount = 10000
	Date_Of_Withdrawal = “2023-01-26”
	PIN_Number = 6324
	Re-inspect Debit Card class

Expected Result	Amount should be withdrawn.
Actual Result	Amount is successfully withdrawn.
Conclusion	The test is successful.

Table 1: Inspecting DebitCard class



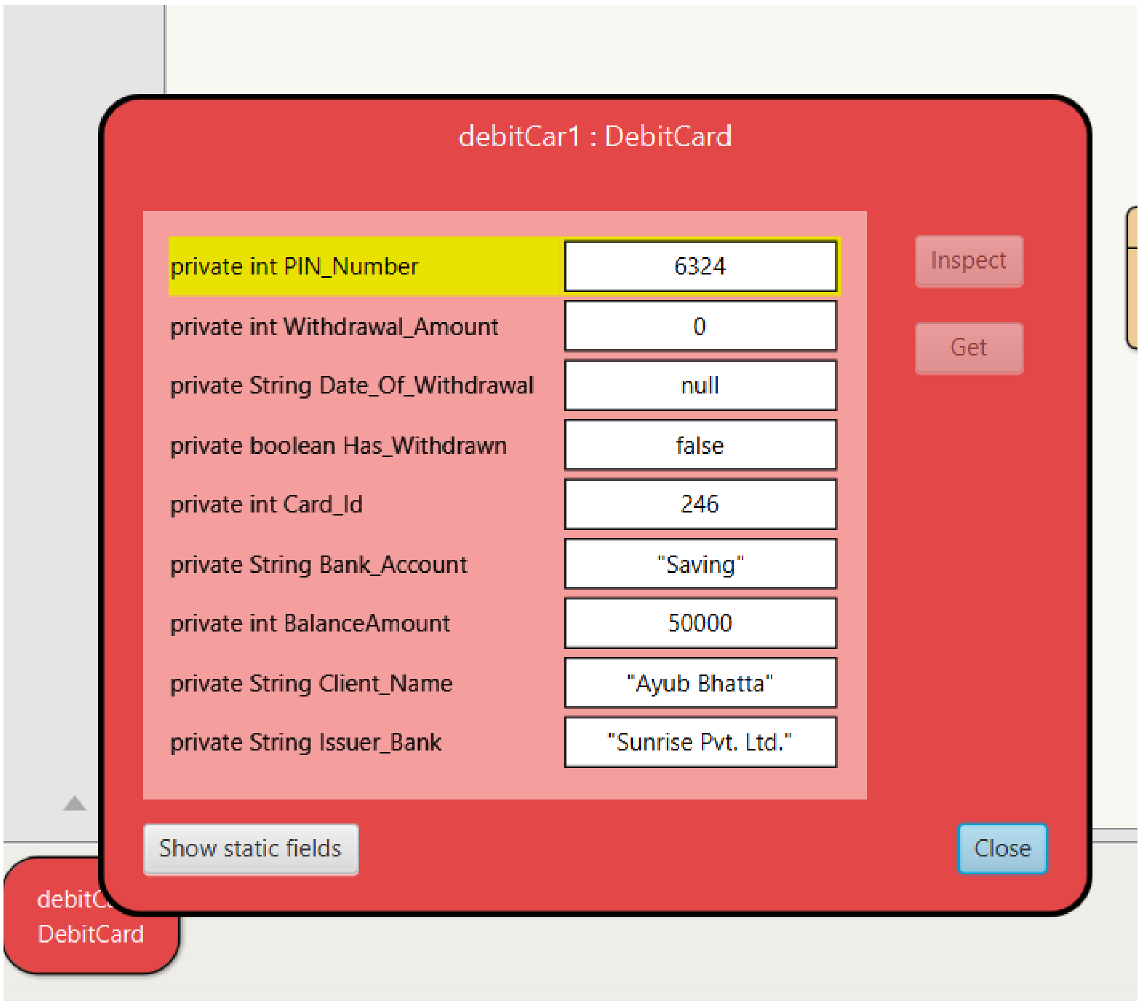
BlueJ: Create Object

DebitCard(int BalanceAmount, int Card_Id, String Bank_Account, String Issuer_Bank, String Client_Name, int PIN_Number)

Name of Instance: debitCar1

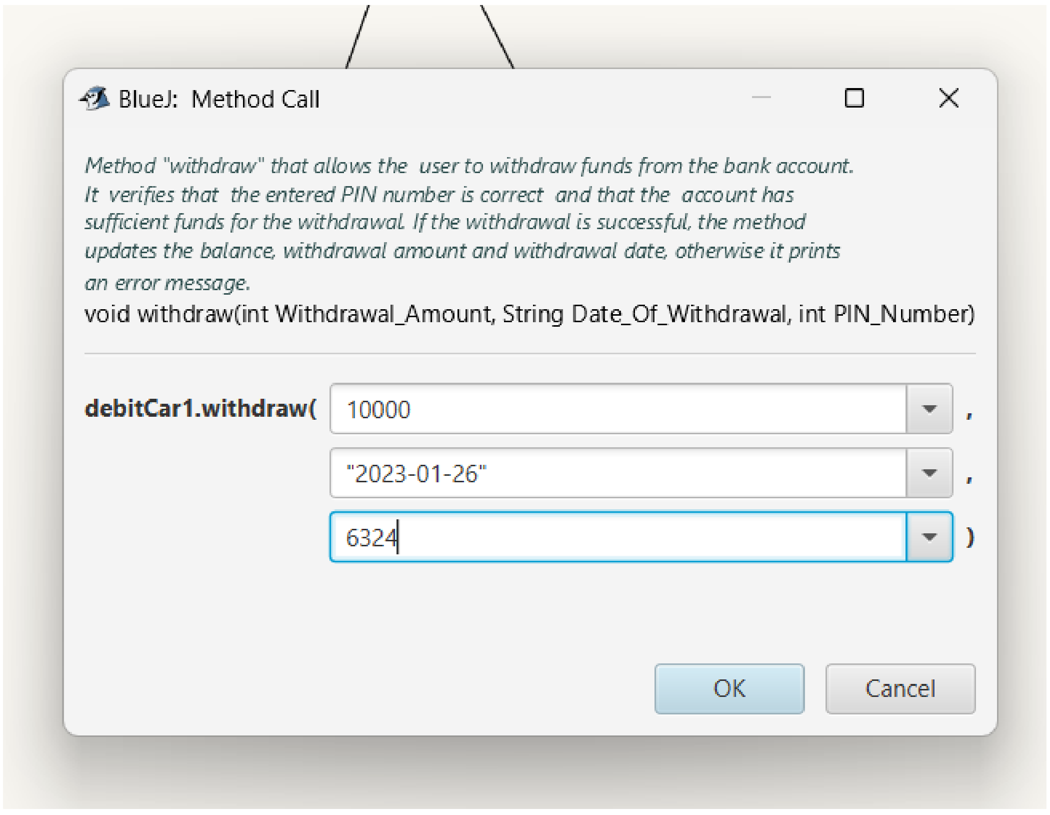
new DebitCard(50000 ,
246 ,
"Saving" ,
"Sunrise Pvt. Ltd." ,
"Ayub Bhatta" ,
6324)

Table-1, Figure 5: Inserting values in DebitCard class



private int PIN_Number	6324
private int Withdrawal_Amount	0
private String Date_Of_Withdrawal	null
private boolean Has_Withdrawn	false
private int Card_Id	246
private String Bank_Account	"Saving"
private int BalanceAmount	50000
private String Client_Name	"Ayub Bhatta"
private String Issuer_Bank	"Sunrise Pvt. Ltd."

Table-1, Figure 6: Inspecting DebitCard class

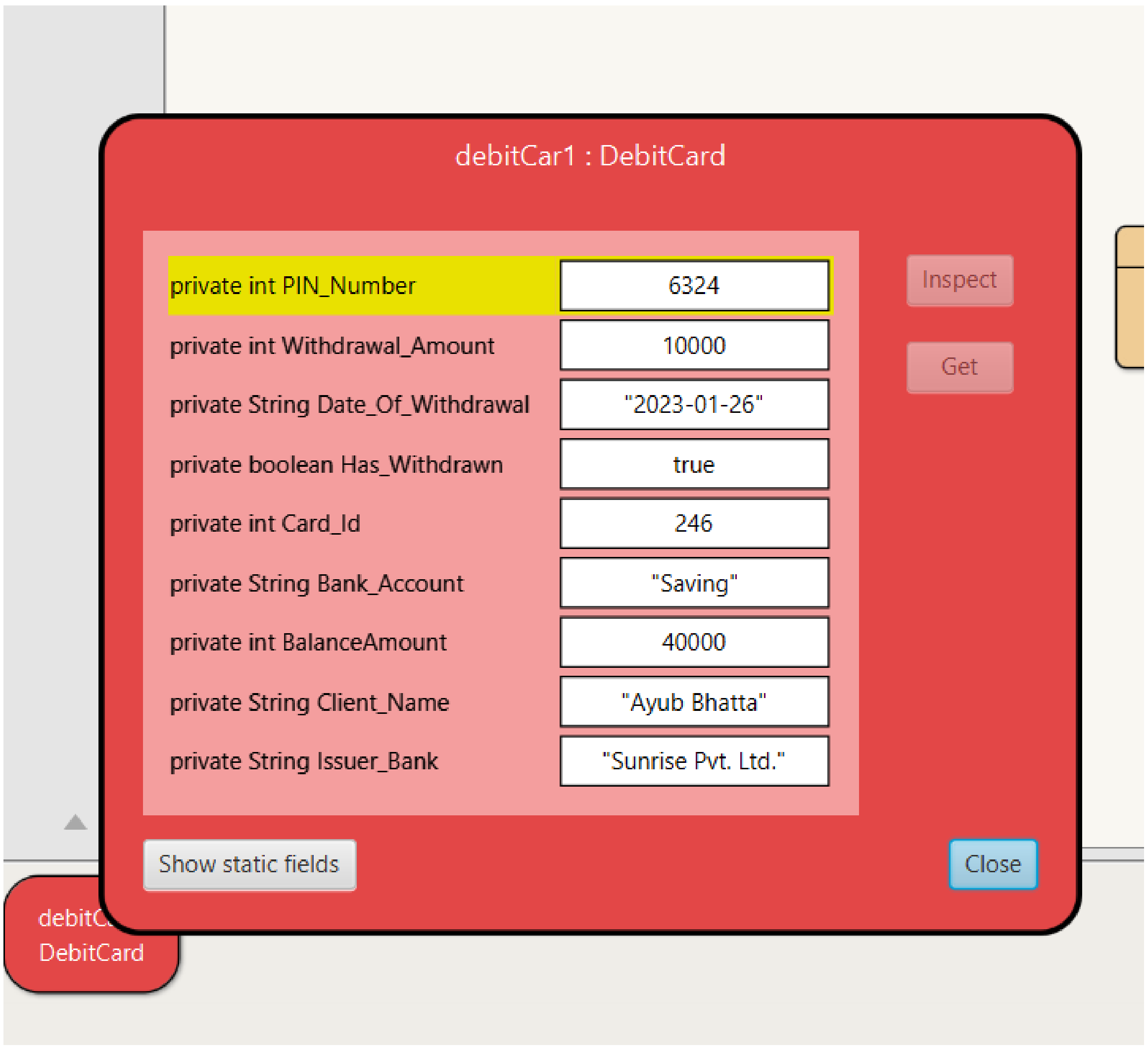

BlueJ: Method Call

Method "withdraw" that allows the user to withdraw funds from the bank account. It verifies that the entered PIN number is correct and that the account has sufficient funds for the withdrawal. If the withdrawal is successful, the method updates the balance, withdrawal amount and withdrawal date, otherwise it prints an error message.

void withdraw(int Withdrawal_Amount, String Date_Of_Withdrawal, int PIN_Number)

debitCar1.withdraw(10000 ,
"2023-01-26" ,
6324)

Table-1, Figure 7: Inserting the values in method Withdraw


debitCar1 : DebitCard

private int PIN_Number	6324
private int Withdrawal_Amount	10000
private String Date_Of_Withdrawal	"2023-01-26"
private boolean Has_Withdrawn	true
private int Card_Id	246
private String Bank_Account	"Saving"
private int BalanceAmount	40000
private String Client_Name	"Ayub Bhatta"
private String Issuer_Bank	"Sunrise Pvt. Ltd."

Table-1, Figure 8: Re-inspection of DebitCard

5.2 Table-2:

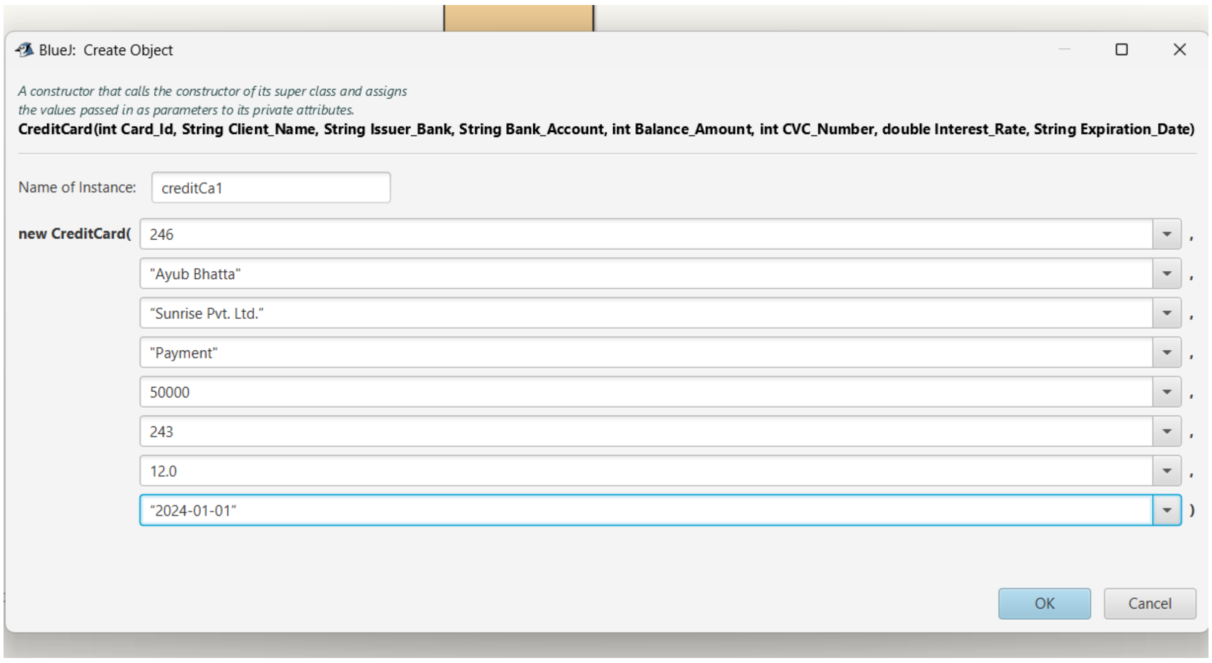
Inspect Credit Card class, set the credit limit and re-inspect the Credit Card Class.

Test-2	
Objective	To inspect Credit Card class, set the Credit Limit and re-inspect the Credit Card class.

Action	Create object of Credit Card class with parameters:
	Card_Id = 246
	Client_Name = “Ayub Bhatta”
	Issuer_Bank = “Sunrise Pvt. Ltd.”
	Bank_Account = “Payment”
	Balance_Amount = 50000
	CVC_Number = 243
	Interest_Rate = 12.0
	Expiration_Date = “2024-01-01”
	Call the set credit limit method of the credit method with parameter
	Credit_Limit = 2.5
	Grace_Period = 5

Expected Result	Credit limit should be set.
Actual Result	Credit limit is set.
Conclusion	The test is successful.

Table 2: Inspecting CreditCard class


BlueJ: Create Object

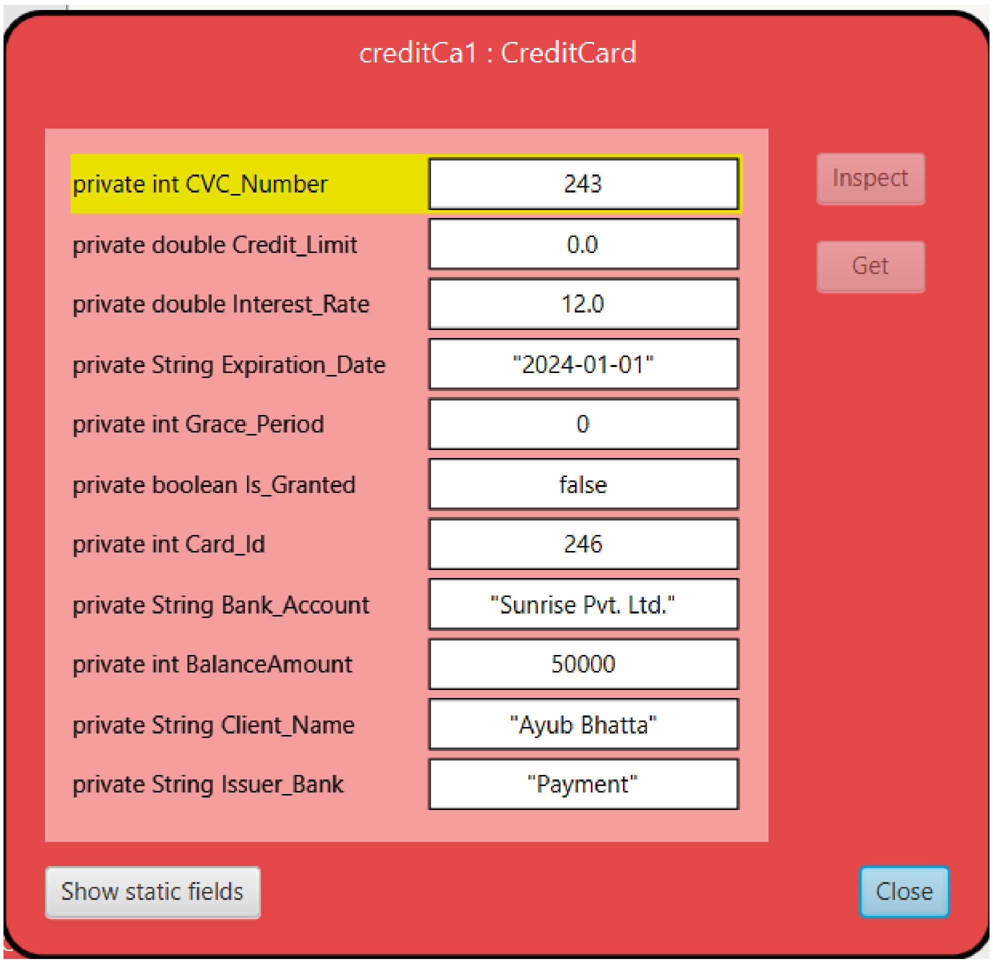
A constructor that calls the constructor of its super class and assigns the values passed in as parameters to its private attributes.

CreditCard(int Card_Id, String Client_Name, String Issuer_Bank, String Bank_Account, int Balance_Amount, int CVC_Number, double Interest_Rate, String Expiration_Date)

Name of Instance: creditCa1

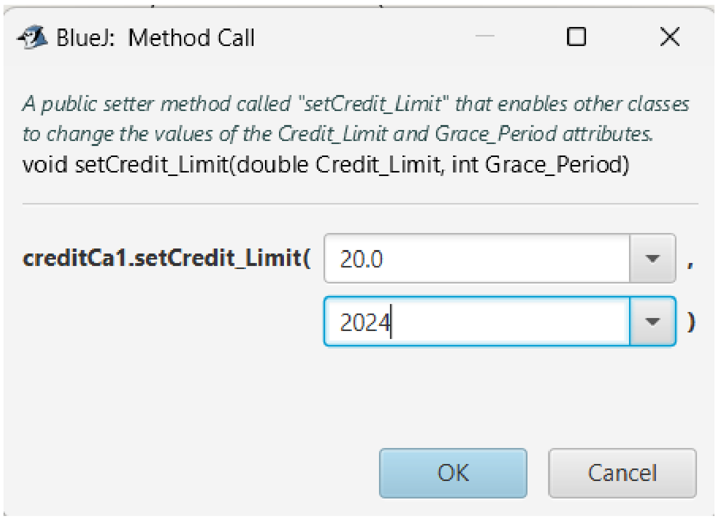
new CreditCard(246 ,
"Ayub Bhatta" ,
"Sunrise Pvt. Ltd." ,
"Payment" ,
50000 ,
243 ,
12.0 ,
"2024-01-01")

Table-2, Figure 9: Inserting the values in CreditCard class


creditCa1 : CreditCard

private int CVC_Number	243
private double Credit_Limit	0.0
private double Interest_Rate	12.0
private String Expiration_Date	"2024-01-01"
private int Grace_Period	0
private boolean Is_Granted	false
private int Card_Id	246
private String Bank_Account	"Sunrise Pvt. Ltd."
private int BalanceAmount	50000
private String Client_Name	"Ayub Bhatta"
private String Issuer_Bank	"Payment"

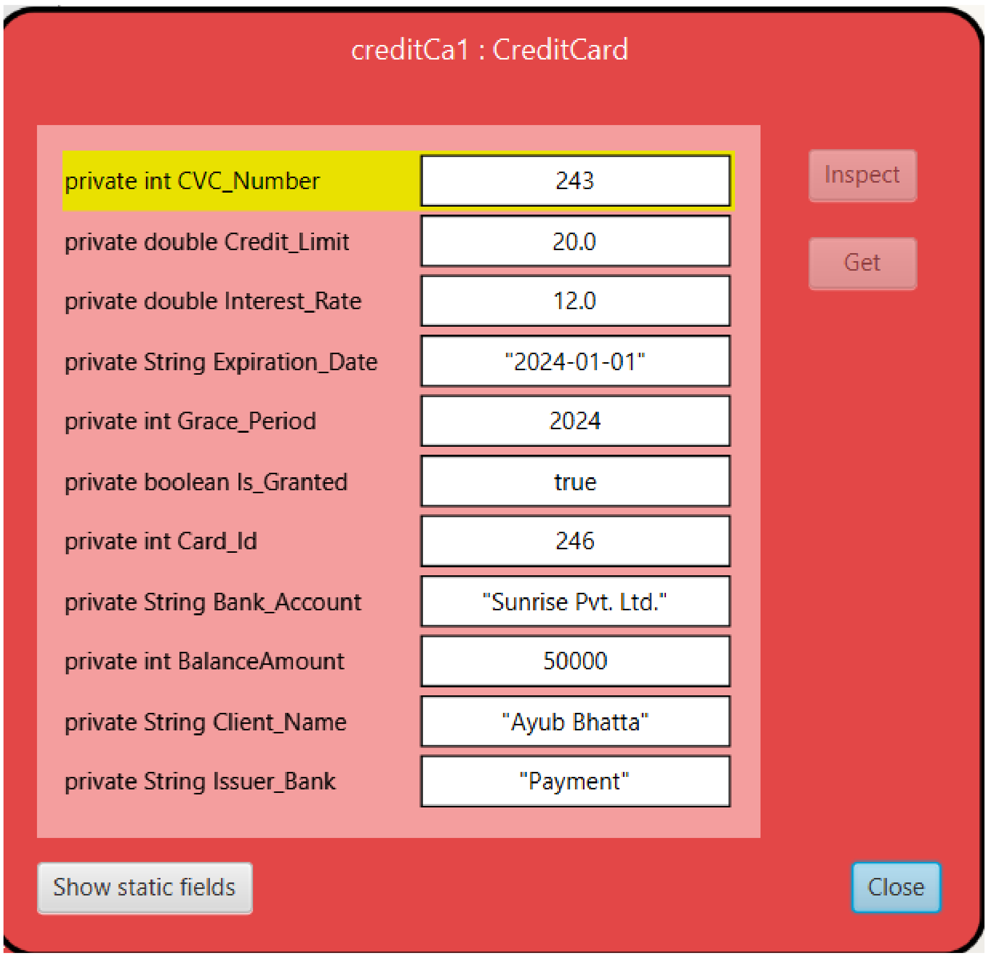
Table-2, Figure 10: Inspecting the CreditCard class


BlueJ: Method Call

A public setter method called "setCredit_Limit" that enables other classes to change the values of the Credit_Limit and Grace_Period attributes.
void setCredit_Limit(double Credit_Limit, int Grace_Period)

creditCa1.setCredit_Limit(20.0 **,**
2024 **)**

Table-2, Figure 11: Setting the Credit_Limit


creditCa1 : CreditCard

private int CVC_Number	243
private double Credit_Limit	20.0
private double Interest_Rate	12.0
private String Expiration_Date	"2024-01-01"
private int Grace_Period	2024
private boolean Is_Granted	true
private int Card_Id	246
private String Bank_Account	"Sunrise Pvt. Ltd."
private int BalanceAmount	50000
private String Client_Name	"Ayub Bhatta"
private String Issuer_Bank	"Payment"

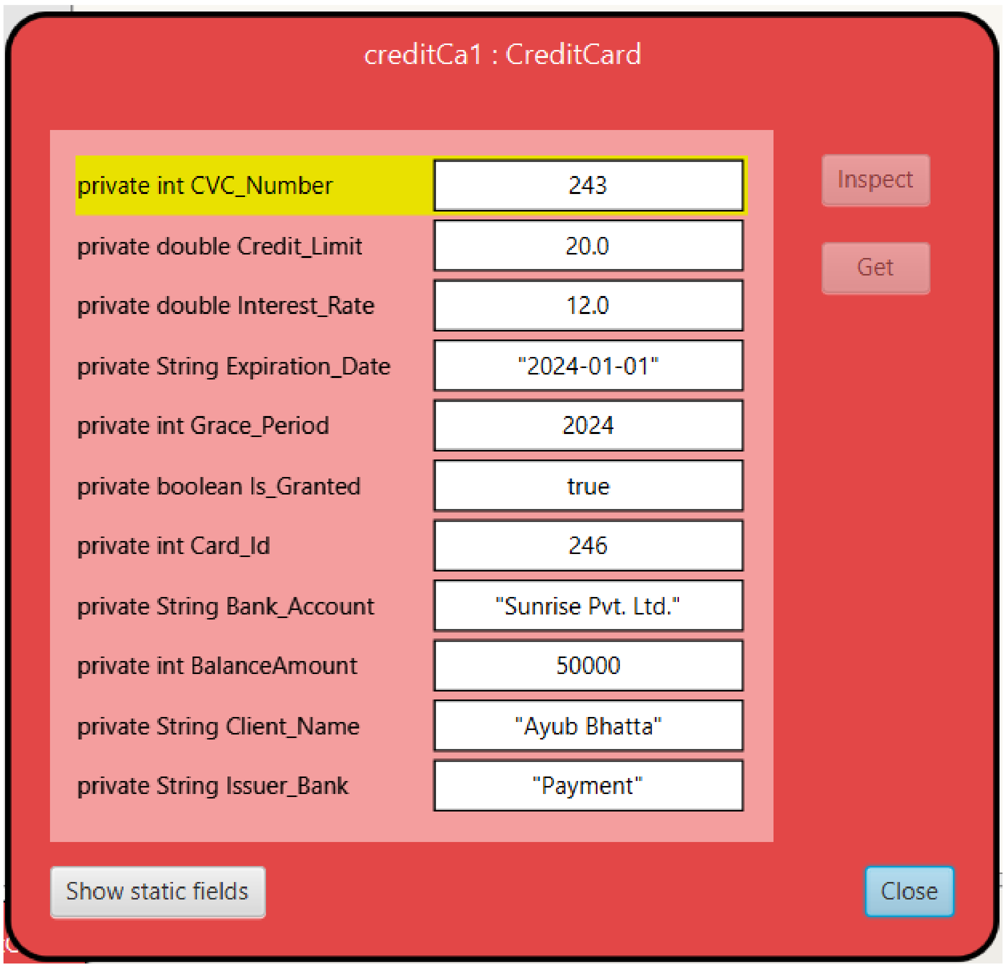
Table-2, Figure 12: Re-inspecting the CredirCard class

5.3 Table-3:

Inspect Credit Card class again after cancelling the credit card.

Test-3	
Objective	To Inspect Credit Card class again after cancelling the credit card.
Action	• Inspecting Credit Card • Cancelling Credit Card • Re-inspecting the Credit Card Class
Expected Result	Credit Card should be Cancelled.
Actual Result	Credit Card is Cancelled.
Conclusion	The test is successful.

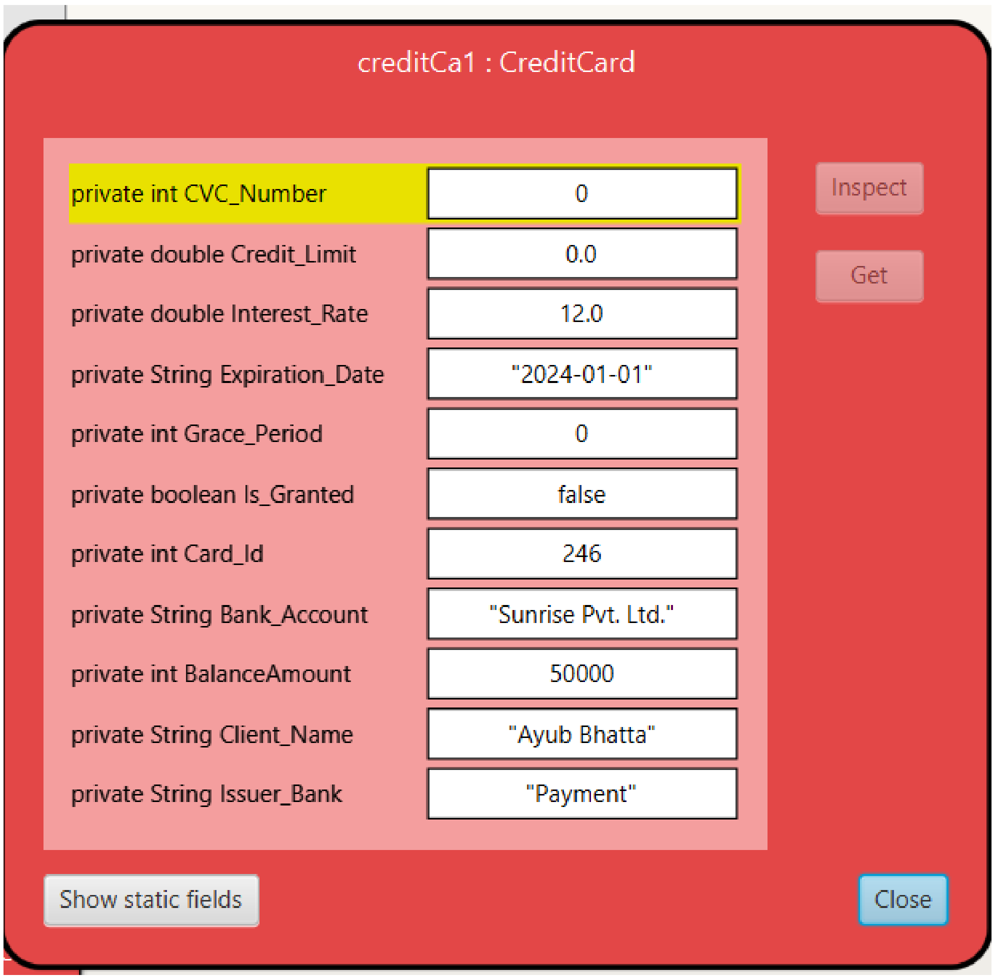
Table 3: Inspecting CreditCard class after Cancelling CreditCard



creditCa1 : CreditCard

private int CVC_Number	243
private double Credit_Limit	20.0
private double Interest_Rate	12.0
private String Expiration_Date	"2024-01-01"
private int Grace_Period	2024
private boolean Is_Granted	true
private int Card_Id	246
private String Bank_Account	"Sunrise Pvt. Ltd."
private int BalanceAmount	50000
private String Client_Name	"Ayub Bhatta"
private String Issuer_Bank	"Payment"

Table-3, Figure 13: Inspecting CreditCard class again


creditCa1 : CreditCard

private int CVC_Number	0
private double Credit_Limit	0.0
private double Interest_Rate	12.0
private String Expiration_Date	"2024-01-01"
private int Grace_Period	0
private boolean Is_Granted	false
private int Card_Id	246
private String Bank_Account	"Sunrise Pvt. Ltd."
private int BalanceAmount	50000
private String Client_Name	"Ayub Bhatta"
private String Issuer_Bank	"Payment"

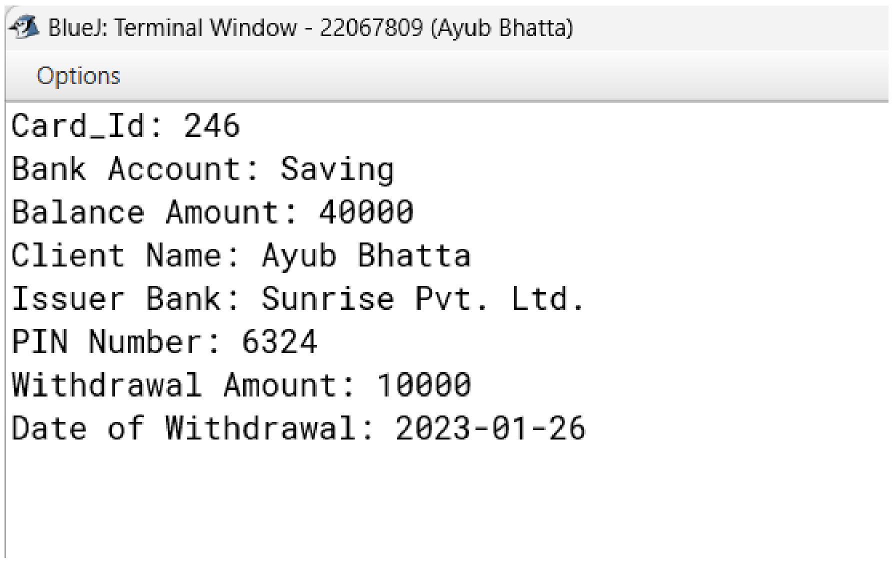
Table-3, Figure 14: Re-inspecting the CreditCard class again after the CancelCreditCard method

5.4 Table-4

Display the details of Debit Card and Credit Card classes.

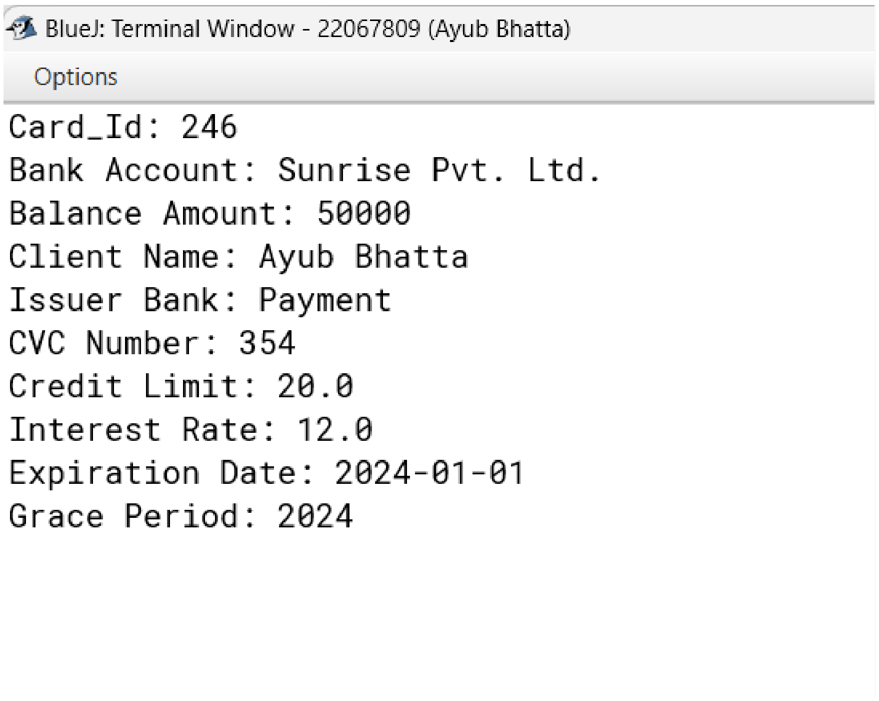
Test-4	
Object	To Display the details of Debit Card and Credit Card classes.
Action	Inspecting and calling the display the method of Debit and Credit Card classes.
Expected Result	The details of Debit and Credit Card classes should be shown below.
Actual Result	The details of Debit and Credit Card classes are shown below.
Conclusion	The test is successful.

Table 4: To display the details of Debit Card and Credit Card classes.



```
Card_Id: 246
Bank Account: Saving
Balance Amount: 40000
Client Name: Ayub Bhatta
Issuer Bank: Sunrise Pvt. Ltd.
PIN Number: 6324
Withdrawal Amount: 10000
Date of Withdrawal: 2023-01-26
```

Figure 15: Void Display method of DebitCard



```
Card_Id: 246
Bank Account: Sunrise Pvt. Ltd.
Balance Amount: 50000
Client Name: Ayub Bhatta
Issuer Bank: Payment
CVC Number: 354
Credit Limit: 20.0
Interest Rate: 12.0
Expiration Date: 2024-01-01
Grace Period: 2024
```

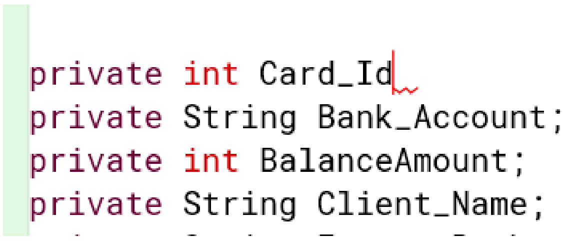
Figure 16: Void Display method of CreditCard

6. Error Detection And Correction

6.1 Error Detection:

6.1.1 Syntax Error:

- The errors which are in our code like spelling errors, punctuation errors, incorrect labels etc are called the syntax errors.

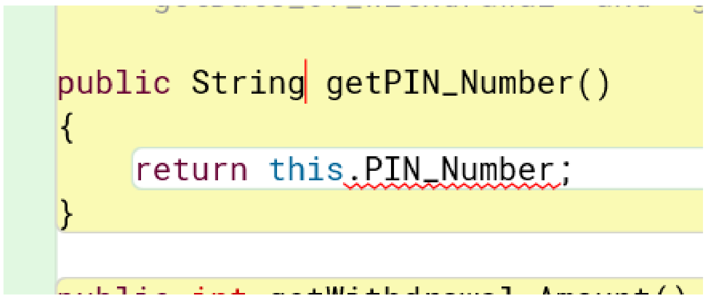


```
private int Card_Id
private String Bank_Account;
private int BalanceAmount;
private String Client_Name;
```

Figure 17: Syntax error

6.1.2 Semantic Error:

- The errors which are in our code which returns the wrong data type to a method are called the semantic errors.

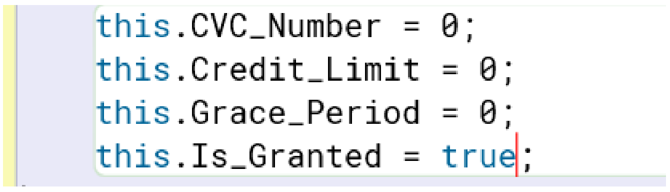


```
public String getPIN_Number()
{
    return this.PIN_Number;
}
```

Figure 18: Semantic error

6.1.3 Logical Error:

- The errors which are in our code like the mathematical errors such that +, -, *, /, etc are called the logical errors.



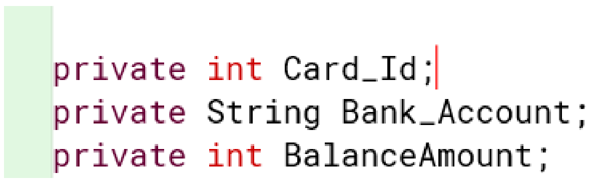
```
this.CVC_Number = 0;
this.Credit_Limit = 0;
this.Grace_Period = 0;
this.Is_Granted = true;
```

Figure 19 Logical error

6.2 Error Correction:

6.2.1 Syntax error correction:

- The syntax error which I had to correct is the missing semicolon (;).

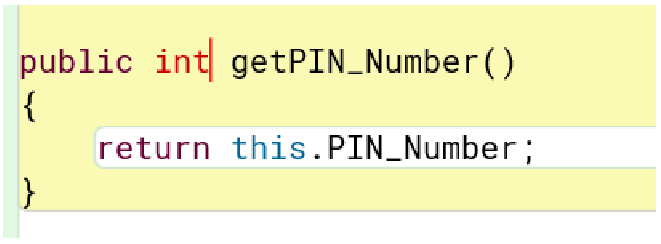


```
private int Card_Id;
private String Bank_Account;
private int BalanceAmount;
```

Figure 20: Syntax error correction

6.2.2 Semantic error correction:

- The semantic error which I had to correct is that I had to change the wrong data type which was String into the correct data type (int) in the accessor method.

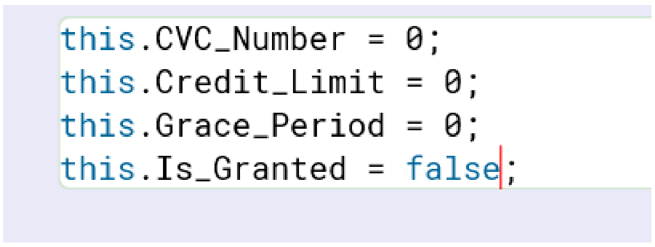


```
public int getPIN_Number()
{
    return this.PIN_Number;
}
```

Figure 21: Semantic error correction

6.2.3 Logical error correction:

- The logical error which I had to correct is that I had to change the Boolean expressing which was True to False as false was the correct Boolean expression needed for the condition.



```
this.CVC_Number = 0;
this.Credit_Limit = 0;
this.Grace_Period = 0;
this.Is_Granted = false;
```

Figure 22: Logical error correction

7. Conclusion

As this is my first project and report as well, I am quite confident that I have done better than I had thought. In this project I had to create a parent class named “Bank Card” with it’s two child classes named as “Debit Card” and “Credit Card”. The parent class and child classes all have their own attributes and their data types. Each have their own constructors, accessor method and mutator method as well. I used getter as the accessor method to get access to the private attributes from the parent class and setter as mutator method to set the values of the given/required instance variables.

As a child class, Debit Card have ‘withdraw’ method which displays the withdrawn amount from the Debit Card class. And Credit Card class have Cancel Credit Card which is used to cancel the Credit Card.

7.1 Evaluation :

As it is seen that I’ve been working with three different classes which are named “Bank Card”, “Debit Card” and “Credit Card” in which the “Bank Card” is the parent class with its own attributes of different data types and some with same data types as well, it also has a constructor and accessor and mutator method as well.

The “Debit Card” class and “Credit Card” class both are the subclasses of “Bank Card” class with its own attributes, constructor and accessor and mutator method as well.

Evaluating for the overall process, the classes are well managed, organised, designed and they provide all the necessary functionalities for the required function of the program. Its better that what I first thought what it would be like before completing my project.

7.2 Concept's Learnt:

In this project, which used Java and Object Oriented Programming (OOP) as its roots to complete our project. I did learned a few things about the concept regarding the advanced use of OOP using Java while completing this project. And the basic concepts I learned are Encapsulation and Inheritance. In this project I had to apply these two concepts for classes, objects and constructors.

8. References

There are no sources in the current document.

9. Appendix

9.1 BankCard class:

```
// The "BankCard" class in this program is used to represent a bank card.

public class BankCard

{

   /* Using several private attributes including the

      client's name, the issuer bank, the card's ID,

      and the the bank account number. */


   private int Card_Id;                    // attribute card_id as an private integer type.

   private String Bank_Account;                    // attribute Bank_Account as a private long
type.

   private int BalanceAmount;                    // attribute BalanceAmount as a int type.

   private String Client_Name;                    // attribute client_name as a private String
type.

   private String Issuer_Bank;                    // attribute Issuer_Bank as a private String
type.


   /*A constructor method that initializes the class

    variables using the parameters passed in.*/


   public BankCard(int BalanceAmount,int Card_Id, String Bank_Account, String
Issuer_Bank)     // Constructor

   {
    // Using this keyword
    this.Client_Name="";
    this.Issuer_Bank=Issuer_Bank;
    this.BalanceAmount=BalanceAmount;
    this.Card_Id=Card_Id;
    this.Bank_Account=Bank_Account;
}


/* Child classes are able to access the values of the private
attributes through the public getter methods such as
"getcard_id", "getBank_Account", "getBalanceAmount",
"getClient_name", and "Issuer_Bank". */
public int getCard_Id()                // getter
{
    return this.Card_Id;
}
public String getBank_Account()            // getter
{
    return this.Bank_Account;
}
public int getBalanceAmount()             // getter
{
  return this.BalanceAmount;
```

```
}
public String getClient_Name()              // getter
{
   return this.Client_Name;
}
public String getIssuer_Bank()              // getter
{
   return this.Issuer_Bank;
}



 /* public setter methods, such as "setClient_name"
  and "setBalanceAmount," that allow other classes
  to change the values of the private attributes. */


public void setClient_name(String Client_Name)              // setter
{
   this.Client_Name=Client_Name;
}


/* A public method "display" which prints the values of the attributes,
 but  only if the "Client_name" attribute  is not empty.  If the
 "Client_name" attribute is empty, the method will print "Error". */
```

```
    public void display()
    {
        if(this.Client_Name.equals(""))
        {
            System.out.println("Error");          // displays "Error" if the client name is empty.
        }

        else
        {
            System.out.println("Card_Id: " + Card_Id);
            System.out.println("Bank Account: " + getBank_Account());
            System.out.println("Balance Amount: " + BalanceAmount);
            System.out.println("Client Name: " + Client_Name);
            System.out.println("Issuer Bank: " + Issuer_Bank);
        }
    }
}
```

9.2 DebitCard class:

```
// The "DebitCard" class in this program extends the "BankCard" class.
public class DebitCard extends BankCard
{
   /* Using several private attributes such as PIN Number,
   Withdrawal Amount, Date Of Withdrawal, and Has Withdrawn */
   private int PIN_Number;                          // attribute PIN_Number as an private integer type.
   private int Withdrawal_Amount;                   // attribute Withdrawal_Amount as an private integer type.
   private String Date_Of_Withdrawal;               // attribute Date_Of_Withdrawal as an private String type.
   private boolean Has_Withdrawn;                   // attribute has_Withdrawan as an private boolean type.



   // constructor calls the constructor of the "BankCard" class with the necessary arguments.
   public DebitCard(int BalanceAmount,int Card_Id, String Bank_Account, String Issuer_Bank, String Client_Name,int PIN_Number)
   {
      // Using super keyword
      super(BalanceAmount, Card_Id, Bank_Account, Issuer_Bank);
```

```
     // Using this keyword
    setClient_name(Client_Name);
    this.PIN_Number=PIN_Number;
    this.Has_Withdrawn=false;
  }

  /*  Other classes can access the values of the private attributes through the
    public getter methods such as "getPIN_Number", "getWithdrawal_Amount",
    "getDate_Of_Withdrawal" and "gethas_Withdrawan" */

  public int getPIN_Number()                                  // getter for attribute
getPIN_Number()
  {
    return this.PIN_Number;
  }

  public int getWithdrawal_Amount()                           // getter for attribute
getWithdrawal_Amount()
  {
    return this.Withdrawal_Amount;
  }

  public String getDate_Of_Withdrawal()                       // getter for attribute
getDate_Of_Withdrawal()
  {
    return this.Date_Of_Withdrawal;
  }

  public boolean getHas_Withdrawn()                    // getter for attribute
getHas_Withdrawan()
  {
    return this.Has_Withdrawn;
  }

  /* A public setter method called "setWithdrawal_amount" that
   allows other classes to change the value of the Withdrawal_Amount attribute. */

  public void setWithdrawal_Amount(int Withdrawal_Amount)                    //
setter for attribute setWithdrawal_Amount
  {
    this.Withdrawal_Amount = Withdrawal_Amount;
  }

  /* Method "withdraw" that allows the  user to withdraw funds from the bank account.
    It  verifies that  the entered PIN number is correct  and that the  account has
    sufficient funds for the withdrawal. If the withdrawal is successful, the method
    updates the balance, withdrawal amount and withdrawal date, otherwise it prints
    an error message.  */
```

```
 public void withdraw(int Withdrawal_Amount, String Date_Of_Withdrawal, int
PIN_Number)
 {
   if (this.PIN_Number == PIN_Number)
   {
     if (Withdrawal_Amount <= getBalanceAmount())
     {
       this.Withdrawal_Amount = Withdrawal_Amount;
       this.Date_Of_Withdrawal = Date_Of_Withdrawal;
       this.Has_Withdrawn = true;
     }

     else
     {
       System.out.println("Insufficient balance!");
     }
   }

   else
   {
     System.out.println("Invalid PIN!");
   }
 }
```

```
    public void display()

    {

        super.display();

        System.out.println("PIN Number: " + getPIN_Number());



        if(Has_Withdrawn)

        {

            System.out.println("Withdrawal Amount: " + getWithdrawal_Amount());

            System.out.println("Date of Withdrawal: " + getDate_Of_Withdrawal());

        }



        else

        {

            System.out.println("BalanceAmount: " + super.getBalanceAmount());

            System.out.println("Pending Transaction.");

        }

    }

}
```

9.3 CreditCard Class:

```
// The "CreditCard" class in this program extends the "BankCard" class.

public class CreditCard extends BankCard

{

  /* Using several private attributes like CVC_Number, Credit_Limit,

   Interest_Rate, Expiration_Date, Grace_Period, and Is_Granted. */


  private int CVC_Number;                // attribute CVC_Number as a private
integer type.

  private double Credit_Limit;           // attribute Credit_Limit as a private
double type.

  private double Interest_Rate;          // attribute Interest_Rate as a private
double type.

  private String Expiration_Date;        // attribute Expiration_Date as a private
String type.

  private int Grace_Period;              // attribute Grace_Period as a private integer
type.

  private boolean Is_Granted;            // attribute Is_Granted as a private boolean
type.


  /*  A constructor that calls the constructor of its super class and assigns

     the values passed in as parameters to its private attributes. */
  public CreditCard(int Card_Id, String Client_Name, String Issuer_Bank, String
Bank_Account, int Balance_Amount, int CVC_Number,
  double Interest_Rate, String Expiration_Date)          // Constructor
  {
    // Using super keyword
    super(Balance_Amount, Card_Id, Issuer_Bank, Bank_Account);


    // Using this keyword
    this.CVC_Number = CVC_Number;
    this.Interest_Rate = Interest_Rate;
    this.Expiration_Date = Expiration_Date;
    setClient_name(Client_Name);
    this.Is_Granted = false;
  }



  /*  Other  classes can  access the values of  the private attributes through  the
    public getter methods such as getCVC_Number, getCredit_Limit, getInterest_Rate,
    getExpiration_Date, getGrace_Period, and getIs_Granted.  */

  public int getCVC_Number()                          // getter
  {
    return this.CVC_Number;
  }
```

```
public double getCredit_Limit()                        // getter
{
   return this.Credit_Limit;
}

public double getInterest_Rate()                        // getter
{
   return this.Interest_Rate;
}

public String getExpiration_Date()                      // getter
{
   return this.Expiration_Date;
}

public int getGrace_Period()                           // getter
{
   return this.Grace_Period;
}

public boolean getIs_Granted()                         // getter
{
   return this.Is_Granted;
```

```
}


/* A public setter method called "setCredit_Limit" that enables other classes

  to change the values of the Credit_Limit and Grace_Period attributes.*/


public void setCredit_Limit(double Credit_Limit, int Grace_Period)

{

   /* This method verifies  if the new  credit limit is less than  or equal

     to 2.5 times the balance amount of the account. If true,it grants the

     credit else, it will print "Credit cannot be issued. Please check your

     account and try again." */


   if(Credit_Limit <= (2.5 * this.getBalanceAmount()))

   {

      this.Credit_Limit = Credit_Limit;

      this.Grace_Period = Grace_Period;

      this.Is_Granted = true;

   }


   else

   {

      System.out.println("Credit cannot be issued. Please check your account and try
again.");

   }
```

```
}



/*  A public method called "cancelCreditCard" that
   allows the user to cancel their credit card. */

public void cancelCreditCard()
{
   /* This method first checks if the credit card  is granted, if true it sets
      the attributes to zero and sets the Is_Granted attribute to false. If the
      credit card is not granted it will print's "Credit card is not active." */
   if(this.Is_Granted)
   {
      this.CVC_Number = 0;
      this.Credit_Limit = 0;
      this.Grace_Period = 0;
      this.Is_Granted = false;
   }

   else
   {
      System.out.println("Credit card is not active.");
   }
}
```

```
    public void display()

    {

        super.display();

        if(this.Is_Granted)

        {

            System.out.println("CVC Number: " + CVC_Number);

            System.out.println("Credit Limit: " + Credit_Limit);

            System.out.println("Interest Rate: " + Interest_Rate);

            System.out.println("Expiration Date: " + Expiration_Date);

            System.out.println("Grace Period: " + Grace_Period);

        }


        else

        {

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

        }

    }

}
```