



**CS4001NI Programming**

**30% Individual Coursework**

**2022-23 Autumn**

**Student Name: Mohammad Nurullah**

**London Met ID: 22067059**

**College ID: NP01NT4A220019**

**Group: L1N1**

**Assignment Due Date: Friday, January 27, 2023**

**Assignment Submission Date: Friday, January 27, 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

[1.1](#_gjdgxs) Introduction 1

[1.1.1](#_30j0zll) Bank Card 1

[1.1.2](#_1fob9te) Debit Card 1

[1.1.3](#_3znysh7) Credit Card 2

[1.1.4](#_2et92p0) Tools Used 2

[1.2](#_tyjcwt) Class Diagram 4

[1.2.1](#_3dy6vkm) Class Diagram of Bank\_Card Class 4

[1.2.2](#_4d34og8) Class Diagram of Debit\_Card Class 5

[1.2.3](#_17dp8vu) Class Diagram of Credit\_Card Class 6

[1.2.4](#_26in1rg) Relationship Diagram Between the Classes 7

[1.3](#_35nkun2) Pseudocode 8

[1.3.1](#_1ksv4uv) Bank\_Card Class 8

[1.3.2](#_44sinio) Debit\_Card Class 11

[1.3.3](#_2jxsxqh) Credit\_Card Class 14

[1.4](#_z337ya) Method Description of Each Class 18

[1.4.1](#_3j2qqm3) Bank\_card 18

[1.4.2](#_1y810tw) Debit\_Card 19

[1.4.3](#_4i7ojhp) Credit\_Card 20

[1.5](#_2xcytpi) Test and Inspection 21

[1.5.1 Test One 21](#_1ci93xb)

[1.5.2 Test 2 26](#_3whwml4)

[1.5.3 Test 3 32](#_2bn6wsx)

[1.5.4 Test 4 36](#_qsh70q)

[1.6](#_3as4poj) Error Detection and Correction 39

[1.6.1](#_1pxezwc) Syntax Error and Solution 39

[1.6.2](#_49x2ik5) Semantic Error and Solution 39

[1.6.3](#_2p2csry) Logical Error and Solution 39

[1.7](#_147n2zr) Conclusion 39

[1.8](#_3o7alnk) Appendix 39

[Figure 1: Class diagram of bank\_card 4](#_1t3h5sf)

[Figure 2: Class diagram of debit card 6](#_2s8eyo1)

[Figure 3: Class diagram of credit card 7](#_3rdcrjn)

[Figure 4: Relationship diagram of classes 8](#_lnxbz9)

Table of tables

# Introduction

The goal of this project is to use Java's object-oriented idea to construct a real-world problem situation. This includes developing a class to represent a bank card and  it’s two subclasses to represent a debit card and a credit card respectively. This program contain three classes a bank card , a debit card and a credit card. Here, bank card represented as parent class and the remaining two classes debit card and credit card represented as sub-classes of bank card class.

## Bank Card

The five attributes that make up the BankCard class are CardId, ClientName, IssuerBank, BankAccount, and BalanceAmount. The client name, issuer bank, bank account, CardID, and balance amount are each represented as strings of text and CardID, and balance amount as a number. The client name attribute is initialized to an empty string in the constructor. The balance amount, card number, bank account, and issure bank are initialized in the constructor by being assigned the value of the parameters.There is a corresponding accessor method for each attribute.

## Debit Card

Debit Card is a subclass of Bank Cards class and have four attributes.. PIN number - an integer , Withdrawal Amount - an integer , date Of Withdrawal - a String and hasWithdrawn – Boolean(either true or false). The constructor accepts six parameters: the balance amount, the card ID, the bank account, the issuer bank, the client name, and the PIN number. The superclass constructor is called with four parameters and a setters method: setclientname. In addition, the parameter values for assign,pin number are assigned.

## Credit Card

Credit Card is a subclass of Bank Cards class and have six attributes.

CVC number- an integer , Credit Limit- a double , Interest Rate - a double , Expiration Date - a String of characters , Grace Period - an integer , isGranted – Boolean(either true or false). Card Id, client name, issuerbank, bank account, BalanceAmount, CVC number, Interest rate, and ExpirationDate are the eight parameters accepted by the constructor. The superclass constructor is called with four parameters and an asetter method. Additionally, in the constructor, assign the parameter values for CVC number, Interest rate, and Expiration Date. The isGranted attribute is set to false.

## Tools Used

1. **Blue J**

BlueJ is a Java development environment that is oriented for beginning and intermediate Java developers and allows developers to easily write Java programs thanks to a simple interface optimized for training.

**Feature of Blue J**

• IDE interface

• Runs on multiple platforms

• Designed for teaching

• Runs without USB stick installation

• Interaction among objects

• Easy to understand and code

1. **Draw.io Diagram**

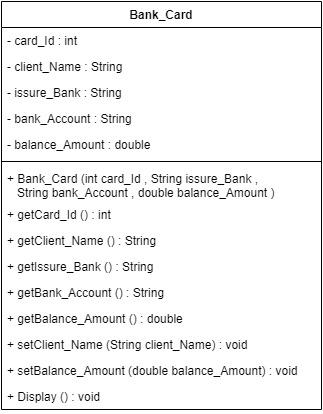
A free, open-source Windows application called Draw.io enables us to make offline or online diagrams. Depending on the type of chart, we can save or import them into other programs or our software. Together with others, we may create workflows that are more efficient.

1. **Microsoft Word**

A word processing tool called Microsoft Word was created by the company for the first time in 1983. It can be found almost anywhere. Many PCs come with Word preinstalled, and it works well with other Microsoft Office applications. Word allows us to construct basic or fancy word processing documents like letters and reports. You may add color, utilize a variety of fonts, and more.

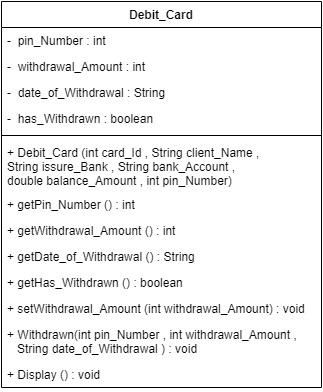
# Class Diagram

## Class Diagram of Bank\_Card Class



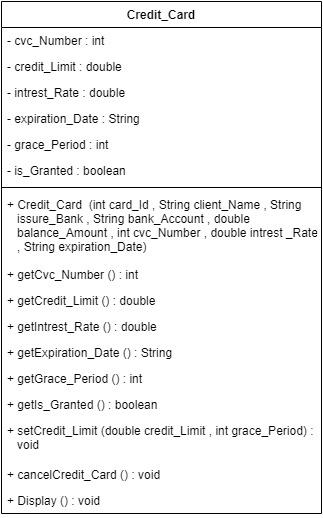
*Figure 1: Class diagram of bank\_card*

## Class Diagram of Debit\_Card Class



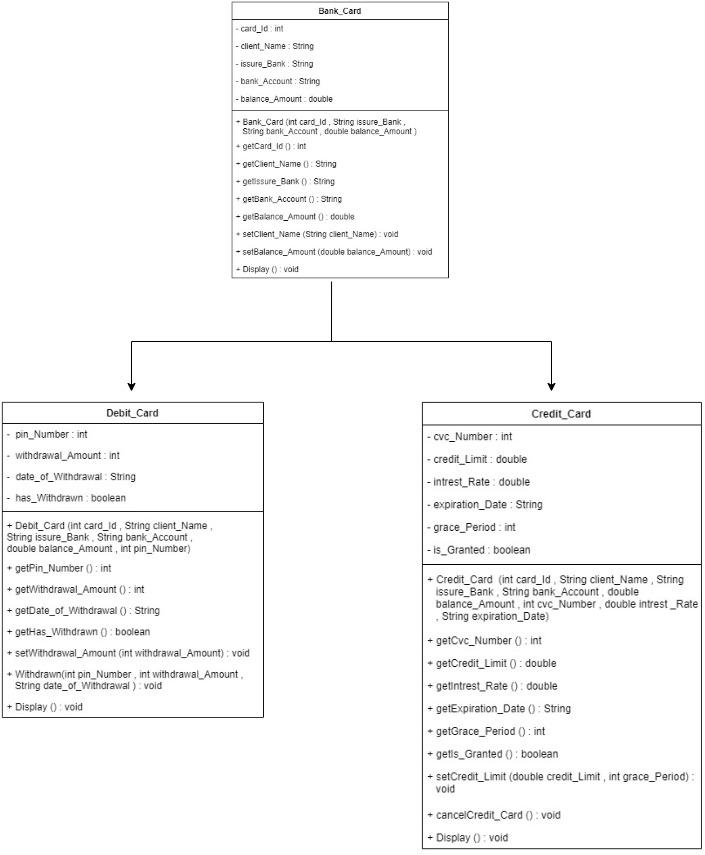
*Figure 2: Class diagram of debit card*

## Class Diagram of Credit\_Card Class



*Figure 3: Class diagram of credit card*

## Relationship Diagram Between the Classes



*Figure 4: Relationship diagram of classes*

# Pseudocode

## Bank\_Card Class

**CREATE** parent class Bank\_Card

**DO**

**DECLARE** instance variable card\_Id with int data type

**DECLARE** instance variable client\_Name with string data type

**DECLARE** instance variable issure\_Bank with string data type

**DECLARE** instance variable bank\_Account with String data type

**DECLARE** instance variable balance\_Ammount with double data type

**CREATE** a constructor Bank\_Card with four parameters card\_Id, issure\_Bank,

bank\_Account, balance\_Amount having data type int, String, String, double

respectively

**DO**

**ASSIGN** the value of instance variable card\_Id from parameter card\_Id

**ASSIGN** the value of instance variable client\_Name as "empty"

**ASSIGN** the value of instance variable issure\_Bank from parameter issure\_Bank

**ASSIGN** the value of instance variable bank\_Account from parameter

bank\_Account

**ASSIGN** the value of instance variable balance\_Amount from parameter

balance\_Amount

**END DO**

**CREATE** an accessor method getcard\_Id with int return data type

**DO**

**RETURN** the value of instance variable card\_Id from parameter card\_Id

**END DO**

**CREATE** an accessor method getclient\_Name with String return data type

**DO**

**RETURN** the value of instance variable client\_Name from parameter

client\_Name

**END DO**

**CREATE** an accessor method getIssure\_Bank with String return data type

**DO**

**RETURN** the value of instance variable issure\_Bank from parameter

issure\_Bank

**END DO**

**CREATE** an accessor method getBank\_Account with String return data type

**DO**

**RETURN** the value of instance variable bank\_Account from parameter

bank\_Account

**END DO**

**CREATE** an accessor method getBalance\_Ammount with double return data type

**DO**

**RETURN** the value of instance variable balance\_Amount from parameter

balance\_Amount

**END DO**

**CREATE** a mutator method setclient\_Name with void return data type with parameter

client\_Name

**DO**

**ASSIGN** the new value to instance variable client\_Name from parameter

client\_Name

**END DO**

**CREATE** a mutator method setBalance\_Amount with void return data type with

parameter balance\_Amount

**DO**

**ASSIGN** the new value to instance variable balance\_Amount from parameter

balance\_Amount

**END DO**

**CREATE** a display method with void return data type

**DO**

**PRINT** the text "Card ID is " and the value of instance variable card\_Id

**PRINT** the text "Issure Bank is " and the value of instance variable

issure\_Bank

**PRINT** the text "Bank account number is " and the value of instance variable

bank\_Account

**PRINT** the text "Account balance is " and the value of instance variable

balance\_Amount

**IF** (client\_Name == "")

**PRINT** the text "Please enter your name "

**END IF**

**ELSE**

**PRINT** the text "Your Name is" and the value of instance variable client\_Name

**END ELSE**

**END DO**

**END DO**

## Debit\_Card Class

**CREATE** a sub class Debit\_Card

**DO**

**DECLARE** instance variable pin\_Number with int data type

**DECLARE** instance variable withdrawal\_Amount with int data type

**DECLARE** instance variable date\_of\_Withdrawal with String data type

**DECLARE** instance variable has\_Withdrawal with boolean data type

**CREATE** a constructor Debit\_Card with six parameters card\_Id, client\_Name,

issure\_Bank, bank\_Account, balance\_Amount, pin\_Number having datatype

int, String, String, String, double, int respectively

**DO**

**CALL** the constructor from parent class Bank\_Card with parameters

**CALL** the mutator setclient\_Name with parameter

**ASSIGN** the value of instance variable pin\_Number from parameter pin\_Number

**ASSIGN** the value of instance variable has\_Withdrawn to false

**END DO**

**CREATE** an accessor method getPin\_Number with int return data type

**DO**

**RETURN** the value of instance variable pin\_Number from parameter pin\_Number

**END DO**

**CREATE** an accessor method getWithdrawal\_Amount with int return data type

**DO**

**RETURN** the value of instance variable withdrawal\_Amount from parameter

withdrawal\_Amount

**END DO**

**CREATE** an accessor method getDate\_of\_withdrawal with String return data type

**DO**

**RETURN** the value of instance variable Date\_of\_Withdrawal from parameter

Date\_of\_Withdrawn

**END DO**

**CREATE** an accessor method getHas\_withdrawn with boolean return data type

**DO**

**RETURN** the value of instance variable has\_Withdrawn from parameter

has\_Withdrawn

**END DO**

**CREATE** a mutator method setWithdrawal\_Amount with void return data type with

parameter withdrawal\_Amount

**DO**

**ASSIGN** the new value to instance variable withdrawal\_Amount from parameter

withdrawal\_Amount

**END DO**

**CREATE** mutator method Withdrawn with the parameters pin\_Number,

withdrawal\_Amount, date\_of\_Withdrawal having data type int,int,String

respectively

**DO**

**IF** (pin\_Number == pin\_Number && withdrawal\_Amount <=

super.getBalance\_Amount()

**ASSIGN** the value of has\_Withdrawn to true

**CALL** setBalance\_Amount(this.getBalance\_Amount()-withdrawal\_Amount)

**ASSIGN** the value of withdrawal\_Amount from the parameter

withdrawal\_Amount

**ASSIGN** the value of date\_of\_Withdrawal from the parameter

date\_of\_Withdrawal

**PRINT** the text "Your amount has withdrawn"

**PRINT** the text "Your withdrawan amount is:" and the value of instance

variable withdrawal\_Amount

**PRINT** the text "Your remaining balance is:" and super.getBalance\_Amount()

**PRINT** the text "Date of withdrawan :" and the value of instance variable

datea\_of\_Withdrawal

**END IF**

**ELSE**

**PRINT** the text "Incorrect PIN Number or Insufficient Account\_Balance "

**END ELSE**

**CREATE** a diplay method with void return type

**DO**

**CALL** the display method of Bank\_Card class

**IF** (has\_Withdrawn == true)

**PRINT** the text “PIN\_Number is” and the value of instance variable

pin\_Number

**PRINT** the text “Withdrawal\_Amount is ” and the value of instance variable

withdrawal\_Amount

**PRINT** the text “Withdrawal-Date is” and the value of instance variable

date\_of\_Withdrawal

**END IF**

**ELSE**

**PRINT** the text "your balance amount is " and the value of getBalance\_Amount

**END ELSE**

**END DO**

**END DO**

## Credit\_Card Class

**CREATE** a sub class Credit\_Card

**DO**

**DECLARE** instance variable cvc\_Number with int data type

**DECLARE** instance variable credit\_Limit with double data type

**DECLARE** instance variable intrest\_Rate with double data type

**DECLARE** instance variable expiration\_Date with String data type

**DECLARE** instance variable grace\_Period with int data type

**DECLARE** instance variable is\_Granted with boolean data type

**CREATE** a constructor Credit\_Card with eight parameters card\_Id, client\_Name,

issure\_Bank, bank\_Account, balance\_Amount, cvc\_Number, intrest\_Rate, expiration\_Date having datatype int, String, String, String, double, int , double, String respectively

**DO**

**CALL** the attributes card\_Id,issure\_Bank,bank\_Account,balance\_Amount

**CALL** the mutator method setClient\_Name

**ASSIGN** the value of cvc\_Number from the parameters cvc\_Number

**ASSIGN** the value of intrest\_Rate from the parameters intrest\_Rate

**ASSIGN** the value of expiration\_Date from the parameters expiration\_Date

**ASSIGN** the value of is\_Granted to false

**CREATE** an accessor method getCvc\_Number with int return data type

**DO**

**RETURN** the value of instance variable cvc\_Number from parameter

cvc\_Number

**END DO**

**CREATE** an accessor method getCredit\_Limit with double return data type

**DO**

**RETURN** the value of instance variable credit\_Limit from parameter

credit\_Limit

**END DO**

**CREATE** an accessor method getIntrest\_Rate with double return data type

**DO**

**RETURN** the value of instance variable intrest\_Rate from parameter

intrest\_Rate

**END DO**

**CREATE** an accessor method getExpiration\_Date with String return data type

**DO**

**RETURN** the value of instance variable expiration\_Date from parameter

expiration\_Date

**END DO**

**CREATE** an accessor method getGrace\_Period with int return data type

**DO**

**RETURN** the value of instance variable grace\_Period from parameter

grace\_Period

END DO

**CREATE** an accessor method getIs\_Granted with boolean return data type

**DO**

**RETURN** the value of instance variable is\_Granted from parameter

is\_Granted

**END DO**

**CREATE** a mutator method setCredit\_Limit with two parameters credit\_Limit,

grace\_Period having data type double, int respectively

**IF** (this.credit\_Limit <= 2\*super.getBalance\_Amount())

**ASSIGN** the value of credit\_Limit from the parameter credit\_Limit

**ASSIGN** the value of grace\_Period from the parameter grace\_Period

**ASSIGN** the value of is\_Granted to true

**PRINT** the text "Congratulation your credit card is on:"

**END IF**

**ELSE**

**PRINT** the text "cannot issue ur credit card"

**END ELSE**

**CREATE** a mutatornmethod cancelCreditCard

**DO**

**ASSIGN** the value of cvc\_Number to 0

**ASSIGN** the value of credit\_Limit to 0

**ASSIGN** the value of grace\_Period to 0

**ASSIGN** the value of is\_Granted to false

**END DO**

**CREATE** a Display method

**DO**

**CALL** the display method of Bank\_Card class

**IF** (is\_Granted == true)

**PRINT** the text "Credit limit is " and the value of an instance variable credit\_Limit

**PRINT** the text "Grace period is " and the value of an instance variable

grace\_Period

**END IF**

**ELSE**

**PRINT** the text "Credit limit is " and the value of an instance variable cvc\_Number

**PRINT** the text "Credit limit is " and the value of an instance variable

intrest\_Rate

**PRINT** the text "Credit limit is " and the value of an instance variable

expiration\_Date

**PRINT** the text "Credit limit is " and the value of an instance variable is\_Granted

**END ELSE**

**END DO**

**END DO**

# Method Description of Each Class

## Bank\_card

| getCard\_Id() | This method is used to return the value of variable card\_Id which is of int data type |
| --- | --- |
| getClient\_Name() | This method is used to return the value of variable client\_Name which is of String data type |
| getIssure\_Bank() | This method is used to return the value of variable issure\_Bank which is of String data type |
| getBank\_Account() | This method is used to return the value of variable bank\_Account which is of String data type |
| getBalance\_Amount() | This method is used to return the value of variable balance\_Amount which is of double data type |
| setClient\_Name() | This method is used to assign the new value to variable client\_Name |
| setBalance\_Amount() | This method is used to assign the new value to variable balance\_Amount |
| Display() | This method is used to display the output card\_Id, issure\_Bank, bank\_Account, balance\_Amount, client\_Name |

*Table 1: Method Discription of Bank\_Card*

## Debit\_Card

| getPin\_Number() | This method is used to return the value of variable pin\_Number which is of int data type |
| --- | --- |
| getWithdrawal\_Amount() | This method is used to return the value of variable withdrawal\_Amount which is of int data type |
| getDate\_of\_Withdrawal() | This method is used to return the value of variable date\_of\_Withdrawal which is of String data type |
| getHas\_Withdrawn() | This method is used to return the value of variable has\_Withdrawn which is of boolean data type |
| setWithdrawal\_Amount() | This method is used to assign the new value to variable withdrawal\_Amount |
| Withdrawn() | This method is used to assign new value to getter method getBalance\_Amount and an instance variable withdrawal\_Amount after performing calculation |
| Display() | This method is used to display the output of the parent class and if it accepts the condition that the  balance has withdrawn then it displays the following output pin\_Number, withdrawal\_Amount, date\_of\_Withdrawal |

*Table 2: Method Description of Debit\_Card*

## Credit\_Card

| getCvc\_Number() | This method is used to return the value of variable cvc\_Number which is of int data type |
| --- | --- |
| getCredit\_Limit() | This method is used to return the value of variable credit\_Limit which is of double data type |
| getIntrest\_Rate() | This method is used to return the value of variable intrest\_Rate which is of double data type |
| getExpiration\_Date() | This method is used to return the value of variable expiration\_Date which is of String data type |
| getGrace\_Period() | This method is used to return the value of variable grace\_Period which is of int data type |
| getIs\_Granted() | This method is used to return the value of variable is\_Granted which is of boolean data type |
| setCredit\_Limit() | This method is used to assign the new value to variable credit\_Limit and grace\_Limit |
| cancelCreditCard() | This method is used to assign the value of cvc\_Number,credit\_Limit,grace\_Period and is\_Granted |
| Display() | This method is used to display the output of the parent class and if it accepts the condition that the transaction is granted then it displays the following output credit\_Limit, grace\_Period, or cvc\_Number, intrest\_Rate, expiration\_Date, is\_Granted |

*Table 3: Method Description of Credit\_Card*

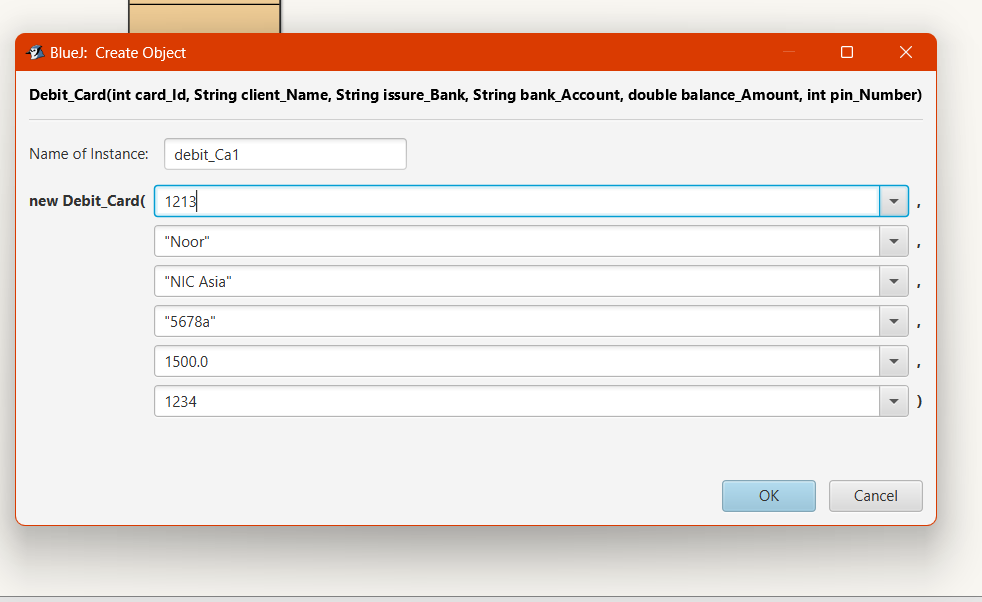
# Test and Inspection

## 1.5.1 Test One

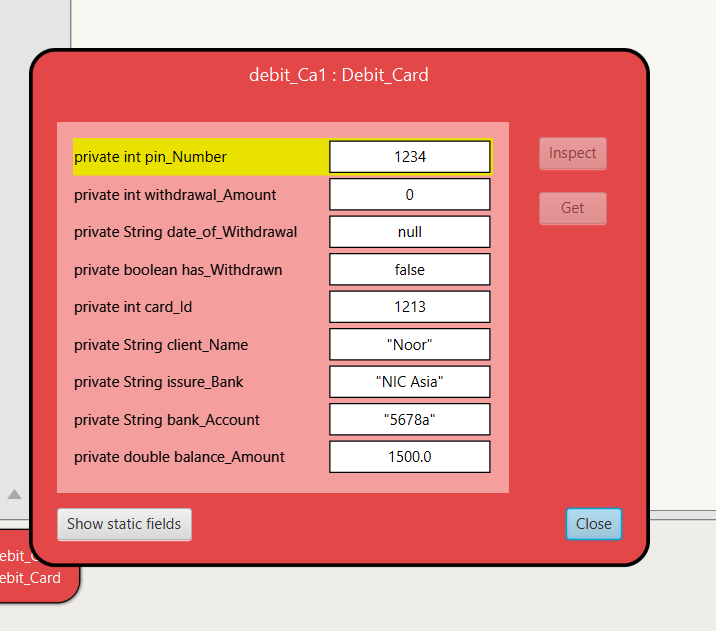
| Test | 1 |
| --- | --- |
| Objective | To Inspect the Debit Card class, withdraw the amount, and re-inspect  the Debit Card Class |
| Action | * The Debit\_Card class is called with the following arguments   card\_Id = 1213  client\_Name = “Noor”  issure\_Bank = “NIC Asia”  bank\_Account = “5678a”  balance\_Amount = 1500.0  pin\_Number = 1234   * Inspection of Debit\_Card was done * Void Withdrawan is called with arguments   pin\_Number = 1234  withdrawal\_Amount = 250  date\_of\_Withdrawal =”25-01-2022”   * Re-inspection of Debit\_Card * Display of void Withdrawan |
| Expected result | The balance will be withdrawn |
| Actual result | The balance is withdrawn |
| Conclusion | The test is successful |

*Table 4: Test 1*

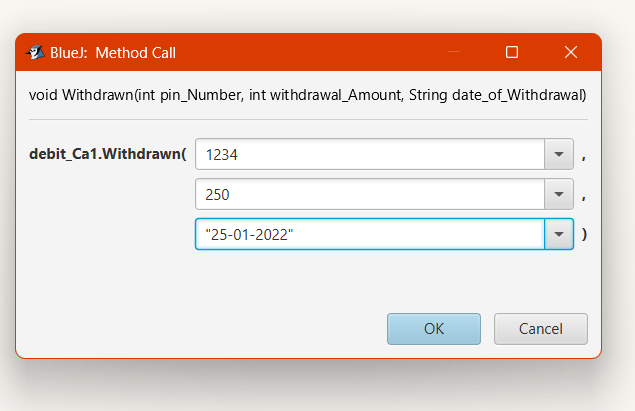
**Output Result:**

****

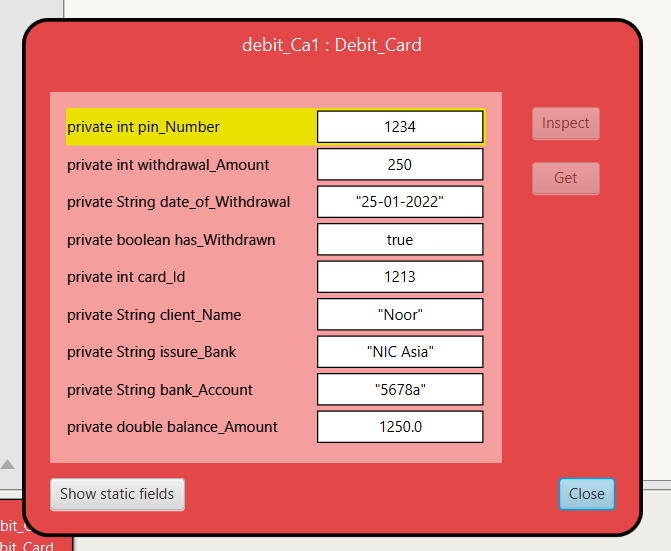
*Figure 5: Assigning the value*

****

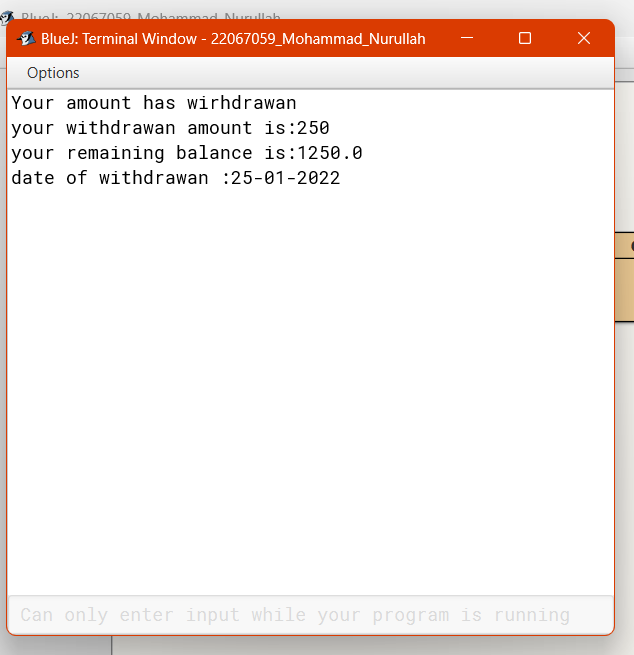
*Figure 6: Inspection*

****

*Figure 7:Assigning the value of void withdrawn*

****

*Figure 8:Re-inspection*

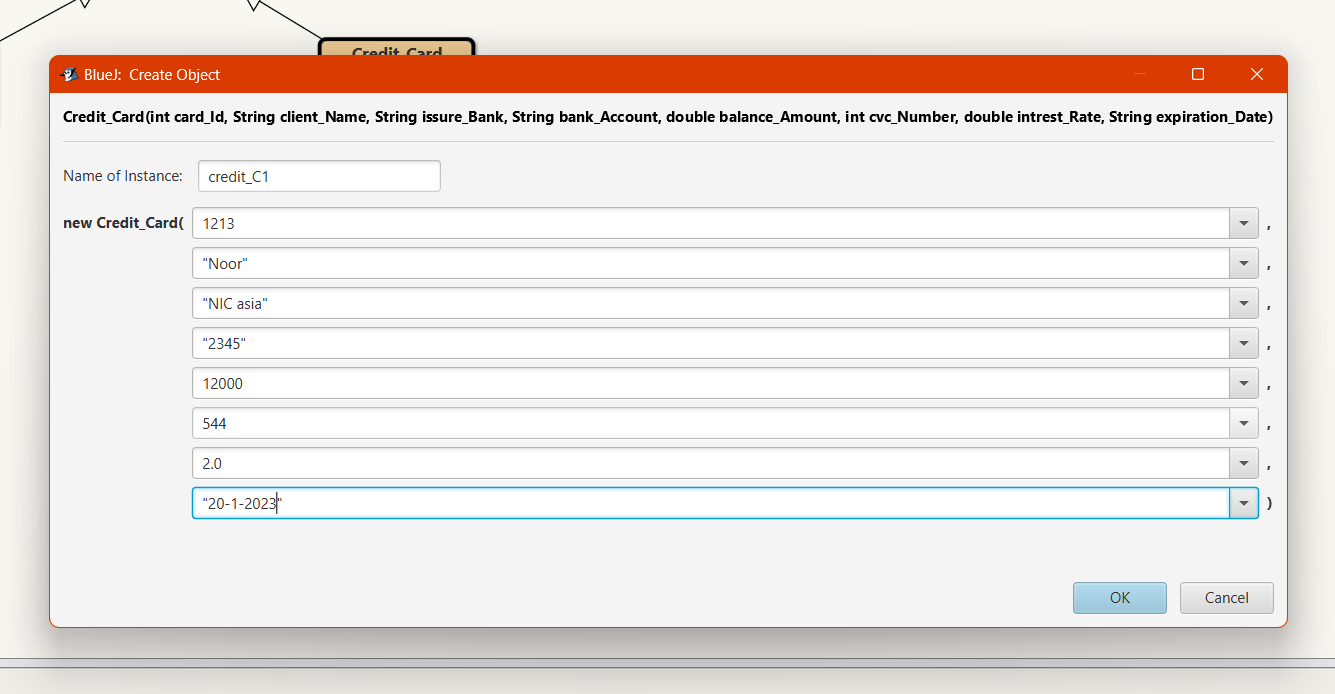
****

*Figure 9: Output Result*

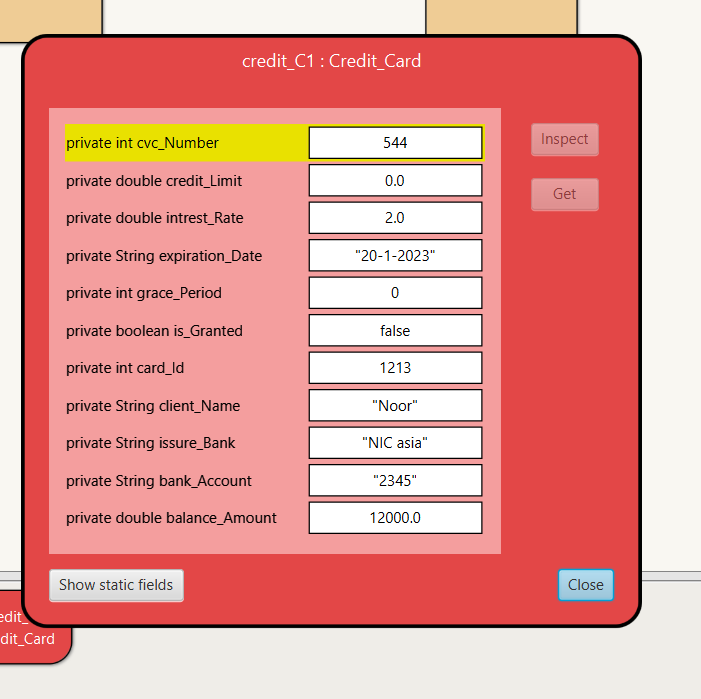
## 1.5.2 Test 2

| Test | 2 |
| --- | --- |
| Objective | To Inspect Credit Card class, set the credit limit and reinspect the  Credit Card class |
| Action | * The Credit\_Card class is called with the following arguments   card\_Id = 1213  client\_Name = “Noor”  issure\_Bank = “NIC asia”  bank\_Account = “2345”  balance\_Amount = 12000  cvc\_Number = 544  intrest\_Rate = 2.0  expiration\_Date = “20-1-2023”   * Inspection of Credit\_Card * Void setCredit\_Limit is called with arguments   credit\_Limit = 4000  grace\_period = 2   * Re-inspection of Credit\_Card * Display of void setCredit\_Limit |
| Expected result | The credit card will be granted |
| Actual result | The credit card is granted |
| Conclusion | The test is successful. |

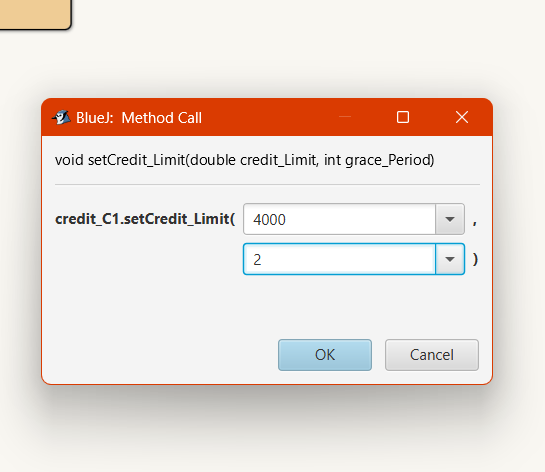
*Table 5: Test 2*

****

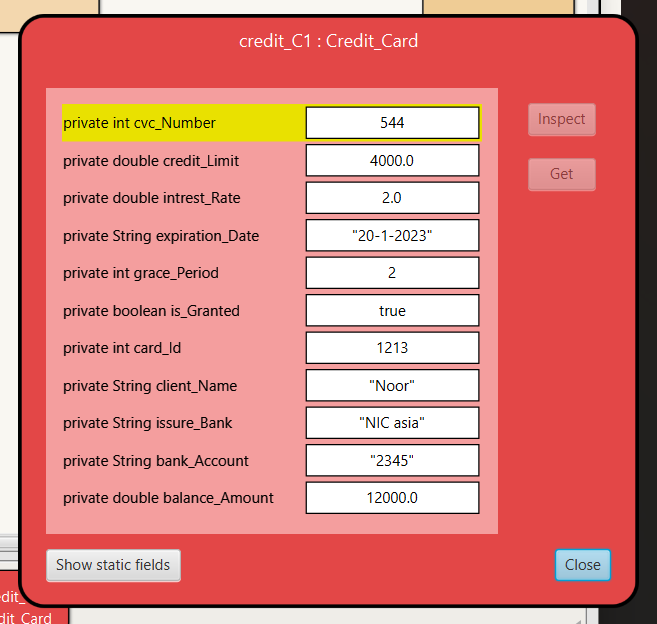
*Figure 10: Assigning the value*

****

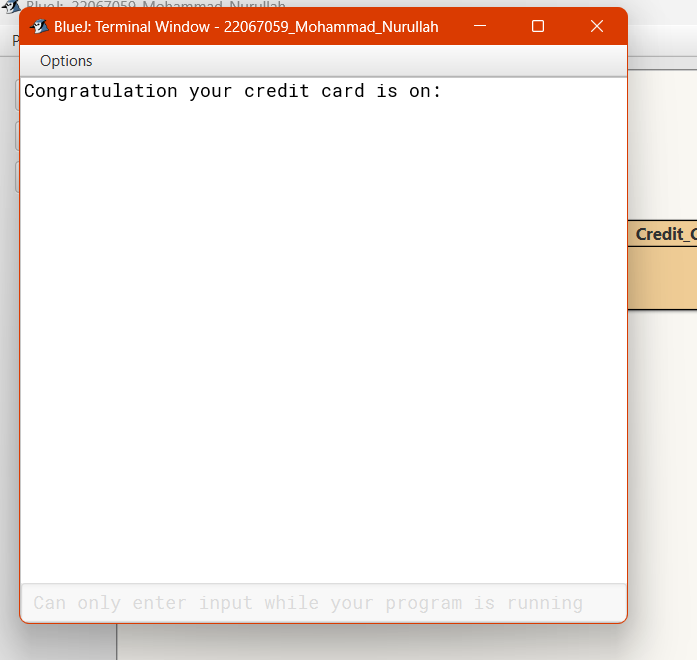
*Figure 11: Inspection*

****

*Figure 12: Assigning the value of void setCredit\_Limit*

****

*Figure 13: Re-inspection*

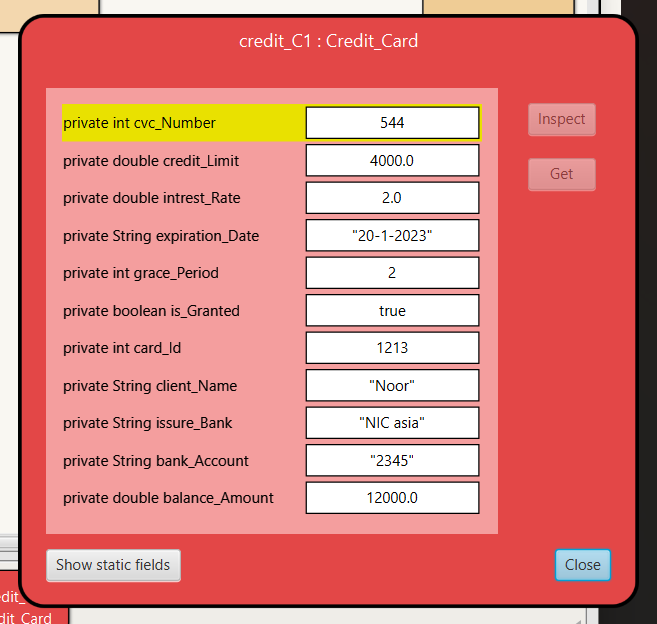
****

*Figure 14: Output Result*

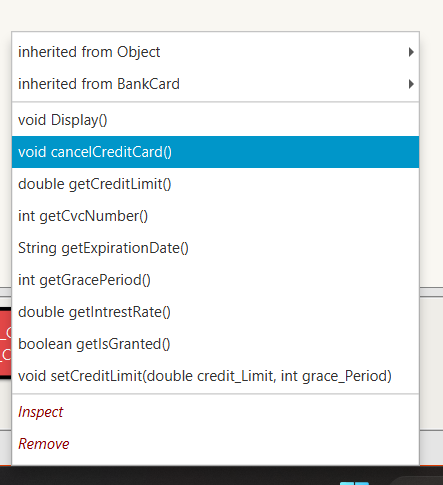
## 1.5.3 Test 3

| Test | 3 |
| --- | --- |
| Objective | To Inspect Credit Card class again after cancelling the credit card. |
| Action | * Inspection of Credit\_Card class is done. * Void cancelCreditCard method is called. * Re-inspection of Credit\_Card class |
| Expected result | The credit card will be cancelled. |
| Actual result | The credit card is cancelled. |
| Conclusion | The test is successful. |

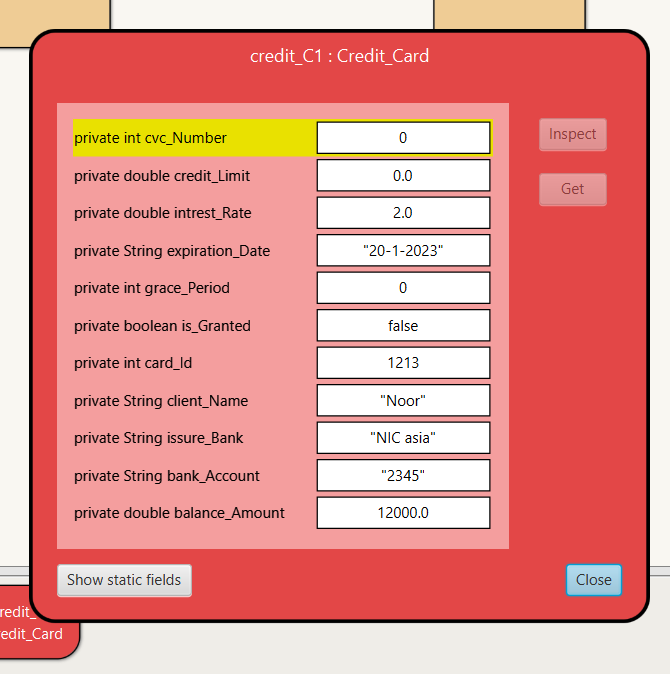
*Table 6: Test 3*



*Figure 15: Inspection*



*Figure 16: Cancelling the method*

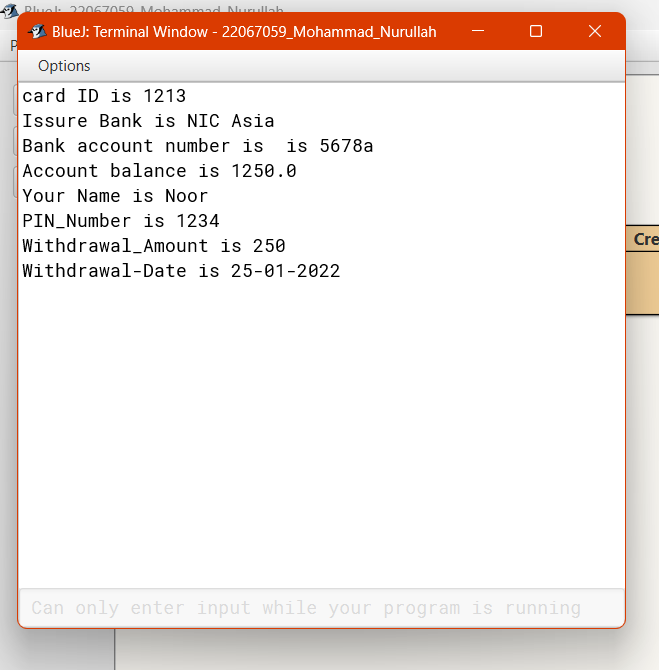


*Figure 17: Re-inspection*

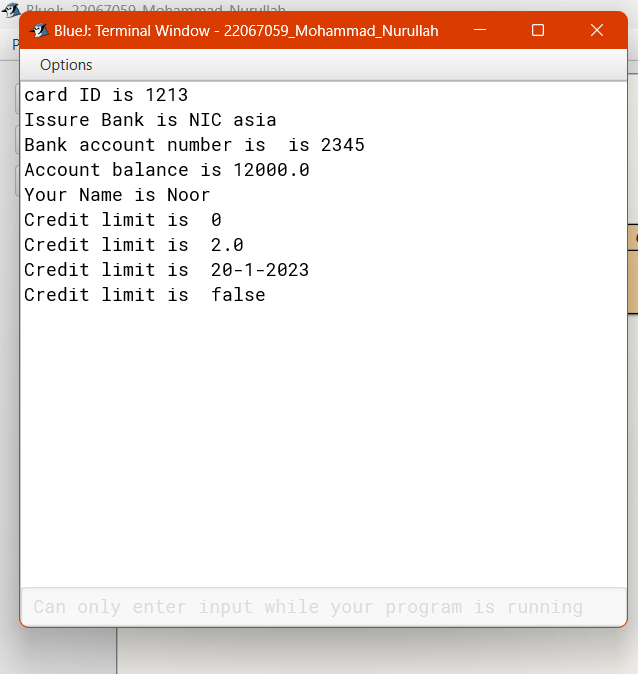
## 1.5.4 Test 4

| Test | 4 |
| --- | --- |
| Objective | To display the details of Debit Card and Credit Card classes. |
| Action | * void Display method of Debit\_Card is called * void Display method of Credit\_Card is called |
| Expected result | It will display the output result of Debit\_Card and Credit\_Card. |
| Actual result | It displayed the output result of Debit\_Card and Credit\_Card. |
| Conclusion | The test is successful |

*Table 7: Test 4*



*Figure 18: Display of Debit\_Card*

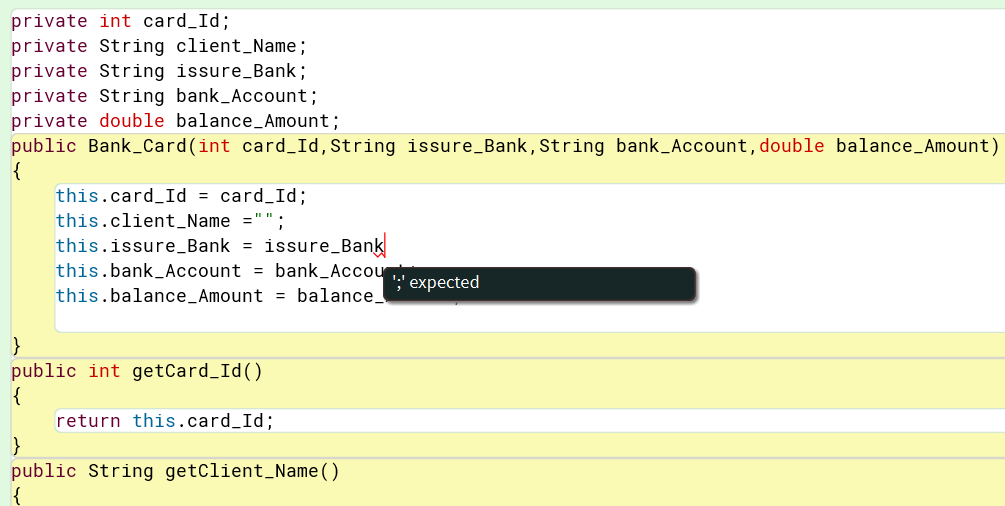


*Figure 19: Display of Credit\_Card*

# Error Detection and Correction

## Syntax Error and Solution

**Screenshot of Syntax error**

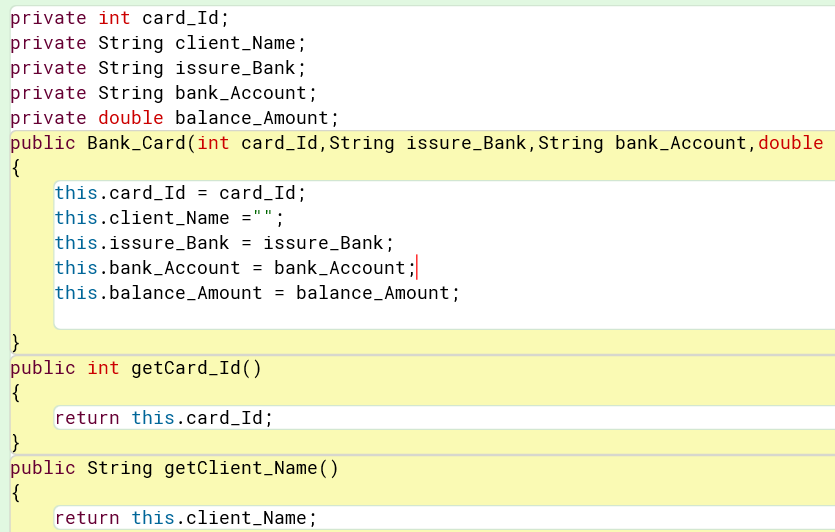


*Figure 20: Screenshot of syntax error*

The structure of different programs is determined by a set of principles known as syntax. Mistakes in source code's syntax include typing errors of instructions, missing to declare variables before using them in programs, and failing to use semicolons after declaring variables.

The semicolon that was missing after declaring the variable in this scenario results in a syntax error. In order to run the code, a semicolon can be added after the declared variable.

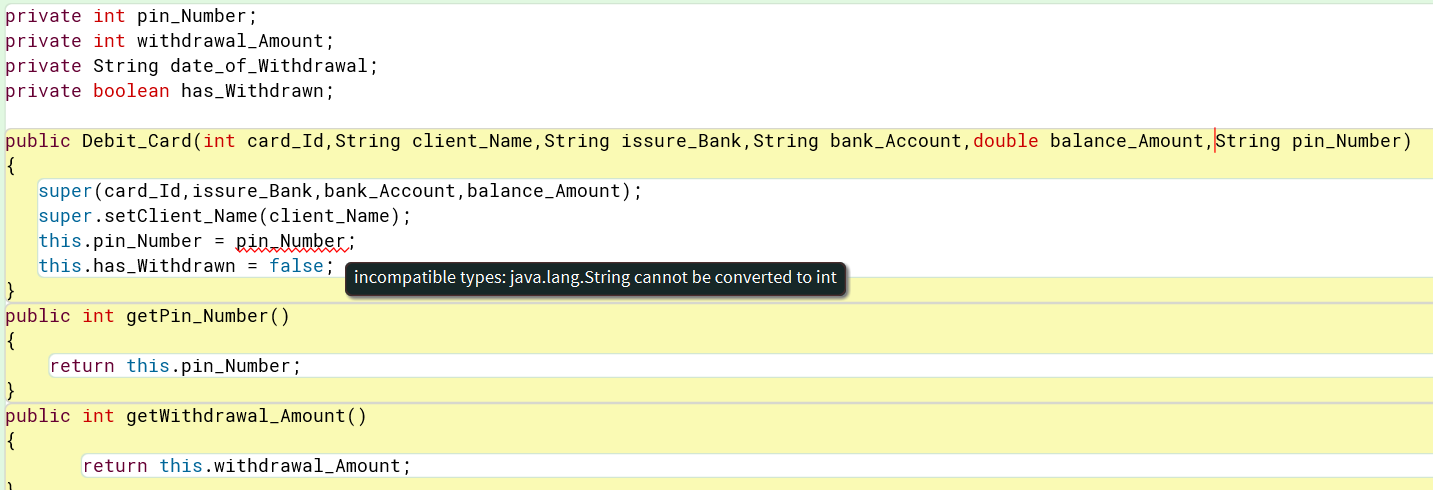
**Screenshot of Syntax error correction**

****

*Figure 21: Screenshot of syntax error correction*

## Semantic Error and Solution

**Screenshot of semantic error**

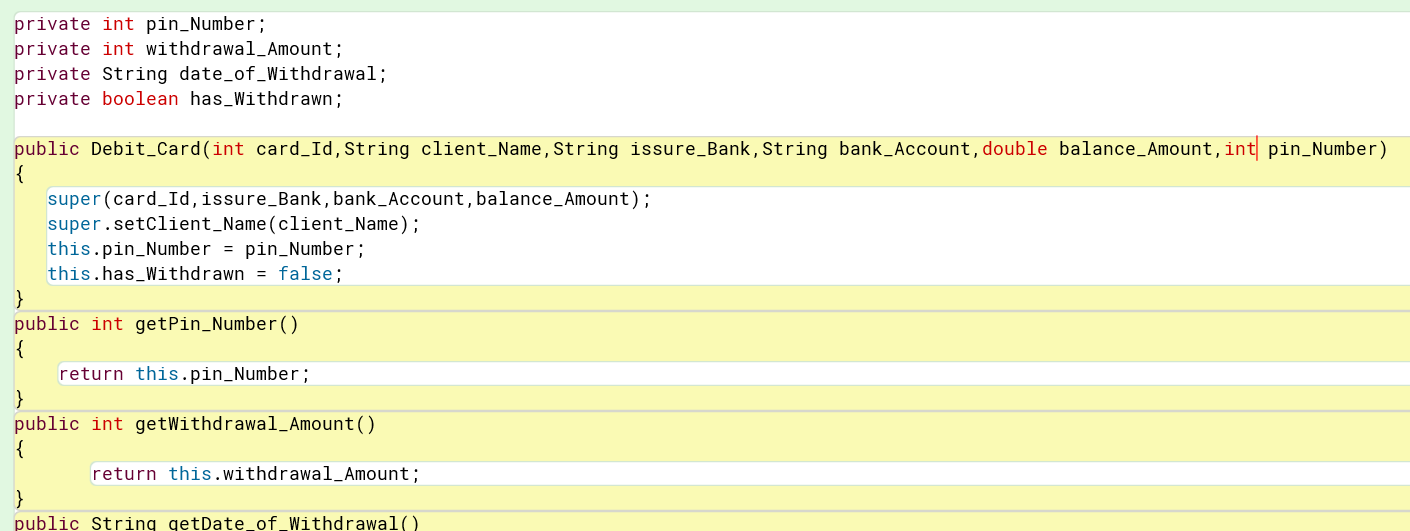
****

*Figure 22: Screenshot of semantic error*

When a statement is syntactically correct but fails to perform the task specified by the user or the compiler, this is known as a semantic error. Some types of basic programming semantic errors are now more easily identified by compilers.

Here, parameter is assigned with a String datatype in the constructor, but the desired outcome requires an Int datatype. Assigning an int datatype to the parameter of constructor will therefore solve the issue. Screenshots that verify the error solution are presented.

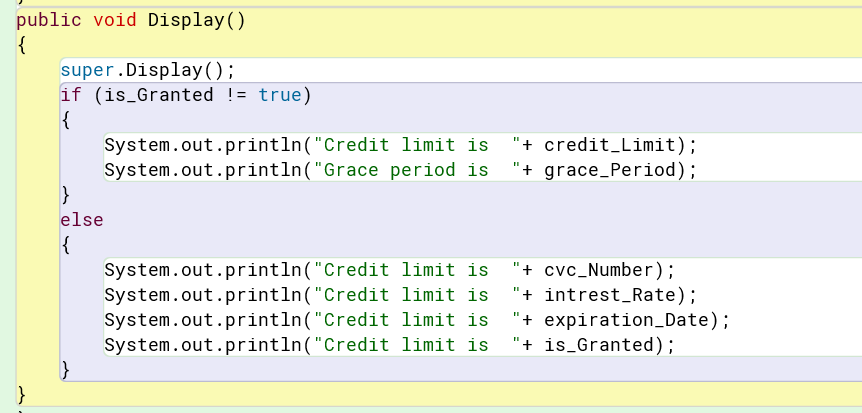
**Screenshot of semantic error correction**

****

*Figure 23: Screenshot of semantic error correction*

## Logical Error and Solution

**Screenshot of logical error**

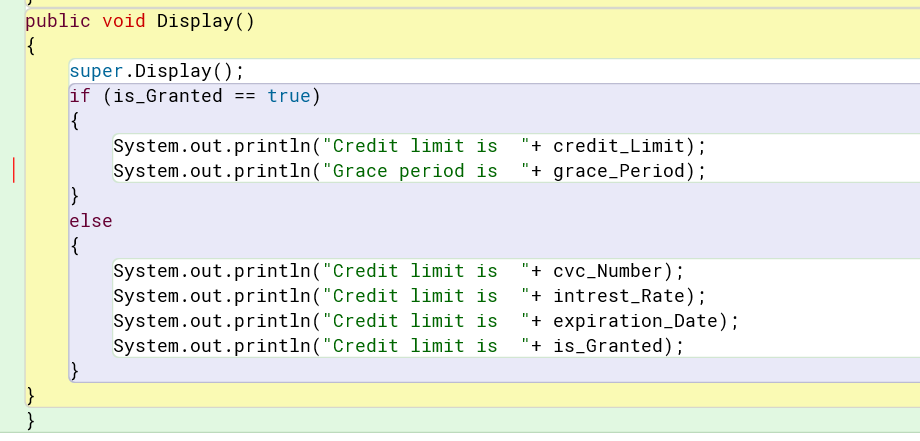
****

*Figure 24: Screenshot of logical error*

A logical error is a programming mistake or error in the source code that results in unexpected output. The condition in the code above determines that is\_Granted is not equal to true and provides an unexpected output based on the users' needs. However, this kind of error is handled by the compiler and does not cause the source code to crash, but it does produce unexpected results.

So, the Equal to Equal operator (==) can be used to correct this logical error. After this, the output is what was anticipated and the if condition checks that is Granted is equal to equal to true. Through screenshots, the error solution is shown.

**Screenshot of logical error correction**

****

*Figure 25: Screenshot of logical error correction*

# Conclusion

The aim of this project is to apply Java's object-oriented concept to execute a real-world problem scenario. To that end, a class called Bank Card and its two subclasses, Debit Card and Credit Card, are being constructed.

This program consists of three classes. Bank, debit, and credit cards. Bank Card is referred to as the parent class or the main class in this case. Subclasses include Debit Card and Credit Card. Also, the parent class inherits both of the subclasses.

We have never performed a task of this kind before. So, during the coursework period, we experienced a lot of misunderstanding and issues. Attending the lecture session and workshop class helped to clarify many concepts. We also gained knowledge in coding testing and debugging. then after, my   Coding and debugging skills improved significantly.

We face significant challenges while we work on this project. We misunderstood the question when the assignment was given, but this issue was resolved by attending the lecture classes. We had no idea how to write pseudocode, but with the help of the internet, we were able to complete it. Thus, managed to overcome all of the challenges and resolving problems, we were able to finish our assignment and submit it in.

# Appendix

## Code of Bank\_Card

public class Bank\_Card

{

private int card\_Id;

private String client\_Name;

private String issure\_Bank;

private String bank\_Account;

private double balance\_Amount;

public Bank\_Card(int card\_Id,String issure\_Bank,String bank\_Account,double balance\_Amount)

{

this.card\_Id = card\_Id;

this.client\_Name ="";

this.issure\_Bank = issure\_Bank;

this.bank\_Account = bank\_Account;

this.balance\_Amount = balance\_Amount;

}

public int getCard\_Id()

{

return this.card\_Id;

}

public String getClient\_Name()

{

return this.client\_Name;

}

public String getIssure\_Bank()

{

return this.issure\_Bank;

}

public String getBank\_Account()

{

return this.bank\_Account;

}

public double getBalance\_Amount()

{

return this.balance\_Amount;

}

public void setClient\_Name(String client\_Name)

{

this.client\_Name = client\_Name;

}

public void setBalance\_Amount(double balance\_Amount)

{

this.balance\_Amount = balance\_Amount;

}

public void Display()

{

System.out.println("card ID is "+ card\_Id);

System.out.println("Issure Bank is "+ issure\_Bank);

System.out.println("Bank account number is is "+ bank\_Account);

System.out.println("Account balance is "+ balance\_Amount);

if(client\_Name == "")

{

System.out.println("Please enter your name");

}

else

{

System.out.println("Your Name is "+ client\_Name);

}

}

}

## Code of Debit\_Card

public class Debit\_Card extends Bank\_Card

{

private int pin\_Number;

private int withdrawal\_Amount;

private String date\_of\_Withdrawal;

private boolean has\_Withdrawn;

public Debit\_Card(int card\_Id,String client\_Name,String issure\_Bank,String bank\_Account,double balance\_Amount,int pin\_Number)

{

super(card\_Id,issure\_Bank,bank\_Account,balance\_Amount);

super.setClient\_Name(client\_Name);

this.pin\_Number = pin\_Number;

this.has\_Withdrawn = false;

}

public int getPin\_Number()

{

return this.pin\_Number;

}

public int getWithdrawal\_Amount()

{

return this.withdrawal\_Amount;

}

public String getDate\_of\_Withdrawal()

{

return this.date\_of\_Withdrawal;

}

public boolean getHas\_Withdrawn()

{

return this.has\_Withdrawn;

}

public void setWithdrawal\_Amount(int withdrawal\_Amount)

{

this.withdrawal\_Amount = withdrawal\_Amount;

}

public void Withdrawn(int pin\_Number,int withdrawal\_Amount,String date\_of\_Withdrawal)

{

if(pin\_Number == pin\_Number && withdrawal\_Amount <= super.getBalance\_Amount())

{

this.has\_Withdrawn = true;

super.setBalance\_Amount(this.getBalance\_Amount()-withdrawal\_Amount);

this.withdrawal\_Amount=withdrawal\_Amount;

this.date\_of\_Withdrawal =date\_of\_Withdrawal;

System.out.println("Your amount has wirhdrawn");

System.out.println("Your withdrawan amount is:" + withdrawal\_Amount);

System.out.println("Your remaining balance is:"+ super.getBalance\_Amount());

System.out.println("Date of withdrawan :" + date\_of\_Withdrawal);

}

else

{

System.out.println("Incorrect PIN Number or Insufficient Account\_Balance ");

}

}

public void Display()

{

super.Display();

if(has\_Withdrawn == true)

{

System.out.println("PIN\_Number is " + pin\_Number);

System.out.println("Withdrawal\_Amount is " + withdrawal\_Amount);

System.out.println("Withdrawal-Date is " + date\_of\_Withdrawal);

}

else

{

System.out.println("your balance amount is "+ getBalance\_Amount());

}

}

}

## Code of Credit\_Card

public class Credit\_Card extends Bank\_Card

{

private int cvc\_Number;

private double credit\_Limit;

private double intrest\_Rate;

private String expiration\_Date;

private int grace\_Period;

private boolean is\_Granted;

public Credit\_Card(int card\_Id,String client\_Name,String issure\_Bank,String bank\_Account,double balance\_Amount,int cvc\_Number,double intrest\_Rate,String expiration\_Date)

{

super(card\_Id,issure\_Bank,bank\_Account,balance\_Amount);

super.setClient\_Name(client\_Name);

this.cvc\_Number = cvc\_Number;

this.intrest\_Rate = intrest\_Rate;

this.expiration\_Date = expiration\_Date;

this.is\_Granted = false;

}

public int getCvc\_Number()

{

return this.cvc\_Number;

}

public double getCredit\_Limit()

{

return this.credit\_Limit;

}

public double getIntrest\_Rate()

{

return this.intrest\_Rate;

}

public String getExpiration\_Date()

{

return this.expiration\_Date;

}

public int getGrace\_Period()

{

return this.grace\_Period;

}

public boolean getIs\_Granted()

{

return this.is\_Granted;

}

public void setCredit\_Limit(double credit\_Limit,int grace\_Period)

{

if(this.credit\_Limit <= 2\*super.getBalance\_Amount())

{

this.credit\_Limit = credit\_Limit;

this.grace\_Period = grace\_Period;

this.is\_Granted = true;

System.out.println("Congratulation your credit card is on:");

}

else

{

System.out.println("cannot issue ur credit card");

}

}

public void cancelCreditCard()

{

this.cvc\_Number = 0;

this.credit\_Limit = 0;

this.grace\_Period = 0;

this.is\_Granted = false;

}

public void Display()

{

super.Display();

if (is\_Granted == true)

{

System.out.println("Credit limit is "+ credit\_Limit);

System.out.println("Grace period is "+ grace\_Period);

}

else

{

System.out.println("Credit limit is "+ cvc\_Number);

System.out.println("Credit limit is "+ intrest\_Rate);

System.out.println("Credit limit is "+ expiration\_Date);

System.out.println("Credit limit is "+ is\_Granted);

}

}

}