



 slington college  
(इस्लिङ्टन कलेज)

## **CS4001NI Programming**

**30% Individual Coursework**

**2022-23 Autumn**

**Student Name: Rojma Shakya**

**London Met ID: 22067659**

**College ID: NP01CP4A220087**

**Group: C4**

**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

Introduction .....	1
I.    Introducing Project Content .....	1
1.    Bank Card .....	1
2.    Debit Card .....	1
3.    Credit Card .....	1
A.    Class Diagram .....	2
B.    Pseudocode .....	3
•    Bank Card .....	3
•    Debit Card .....	6
•    Credit Card .....	10
C.    Method Descriptions .....	14
•    Bank Card .....	14
•    Debit Card .....	14
•    Credit Card .....	14
D.    Testing(Inspection) .....	15
•    Test 1 : To inspect the Debit Card class, withdraw the amount, and re-inspect the Debit Card Class .....	15
Test No: .....	15
Output Result: .....	16

• Test 2 : To Inspect Credit Card class, set the credit limit and reinspect the Credit Card class.....	19
Test No: .....	19
Output Result: .....	20
• Test 3 : To inspect Credit Card class again after cancelling the credit card .....	24
Test No: .....	24
Output Results: .....	25
• Test 4 : To display the details of Debit Card and Credit Card classes.....	27
Test No: .....	27
Output Results: .....	28
E. Different error detection and correction.....	30
○ Syntax errors.....	30
Error detection .....	30
Error correction.....	31
○ Semantic errors.....	32
Error Detection .....	32
Error Correction .....	33
○ Logical Errors.....	34
Error Detection .....	34
Error Correction .....	35
F. Conclusion .....	36
G. References.....	37
H. Appendix.....	38
• Bank Card .....	38

- Debit Card..... 41
- Credit Card..... 45

## Table of Figures

Figure 1:Screenshot of assigning the data in Debit Card class .....	16
Figure 2:Screenshot for the inspection of Debit Card class.....	17
Figure 3: Screenshot of assigning the data in void withdraw in Debit Card class.....	18
Figure 4: Screenshot of assigning the data in Credit Card class.....	20
Figure 5: Screenshot for the inspection of Credit Card class .....	21
Figure 6: Screenshot of assigning the data in void setcreditLimit in Debit Card class..	22
Figure 7: Screenshot for the inspection of void setcreditLimit in Credit Card class .....	23
Figure 8:Screenshot of the output after credit card is cancelled.....	25
Figure 9: Screenshot for the inspection of cancel Credit Card . .....	26
Figure 10:Screenshot of displaying the details of Debit Card.....	28
Figure 11: Screenshot of displaying the details of Credit Card.....	29
Figure 12: Screenshot of displaying the details of cancel Credit Card .....	29
Figure 13:Screenshot of syntax error .....	30
Figure 14:Screenshot of correction of Syntax error.....	31
Figure 15: Screenshot of semantic error .....	32
Figure 16: Screenshot of correction of Semantic error .....	33
Figure 17: Screenshot of logical error .....	34
Figure 18: Screenshot of correction of logical error.....	35

## **Introduction**

### **I. Introducing Project Content**

#### **1. Bank Card**

When users ask for a bank card , banks tells the user to provide them with user's suitable card Id , client name , issuer bank , bank account and their balance Amount . But if the client name is not assigned , it displays a suitable message .

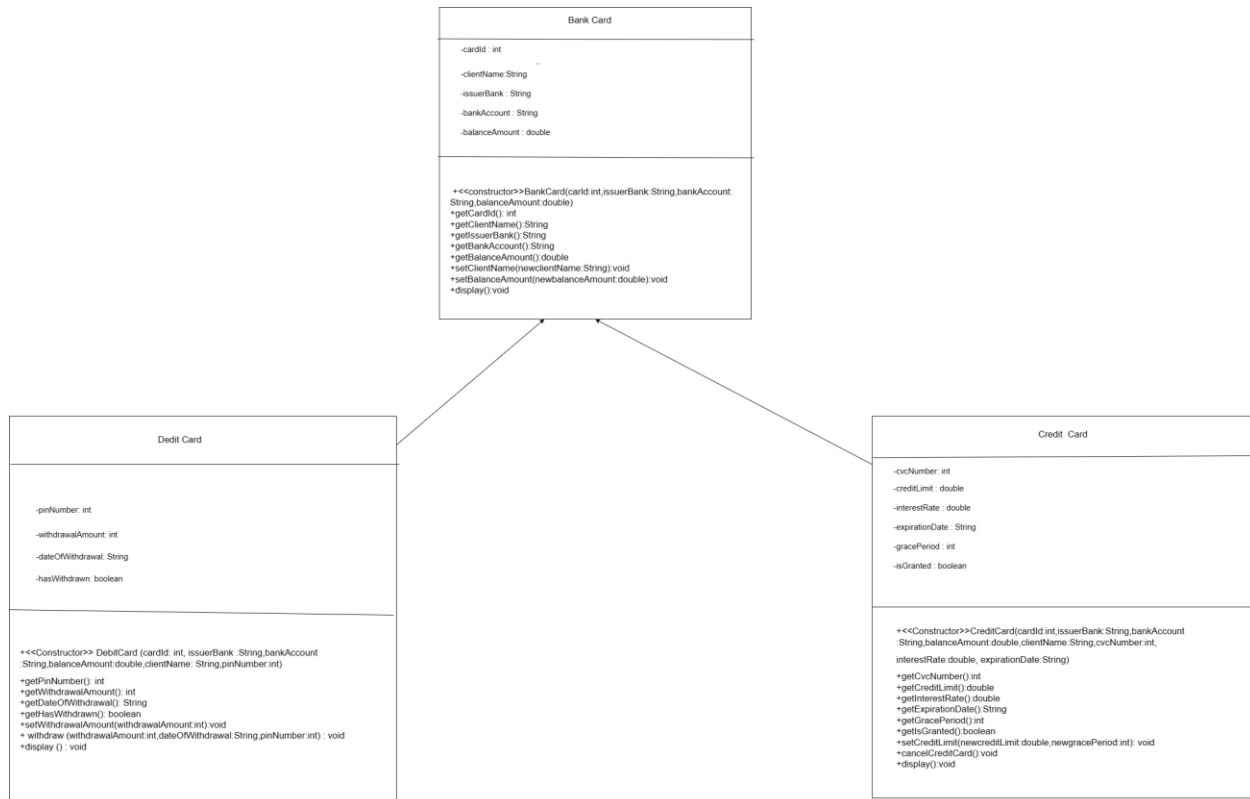
#### **2. Debit Card**

When a user having a Bank Card asks for a debit card, they need to provide the bank with their suitable PIN number, withdrawal amount, date of withdrawal, and if they have withdrawn or not .When a valid pin number is entered and sufficient amount is also present , then only the amount can be withdrawn .But if the user has entered wrong pin number or there is no sufficient balance, then a suitable message is displayed to that user.

#### **3. Credit Card**

When a user having a bank card asks for a credit card,they provide their cvc number, credit limit , interest rate , expiration date , grace period .The credit limit is set as each and every user has different credit limits , then a new credit limit and a new Grace period is accepted .The credit is granted to the user , if the credit limit is less than or equal to 2.5 times the balance amount , but if credit has not been granted to the user then a suitable message is displayed. And the cancelCreditcard will remove the user's credit card .

## A. Class Diagram





## **B. Pseudocode**

- **Bank Card**

**CREATE** a parent class BankCard

**DECLARE** instance variable cardId as int using private modifier

**DECLARE** instance variable clientName as string using private modifier

**DECLARE** instance variable issuerBank as string using private modifier

**DECLARE** instance variable bankAccount as string using private modifier

**DECLARE** instance variable balanceAmount as double using private modifier

**CREATE** a constructor of Bank Card with parameters(cardId, issuerBank, bankAccount, balanceAmount)

**SET** the parameter value for cardId

**SET** the parameter value for clientName

**SET** the parameter value for issuerBank

**SET** the parameter value for bankAccount

**SET** the parameter value for balanceAmount

**CREATE** an accessor method getCardId() with return type int

**DO**

**RETURN** cardId

**END DO**

**CREATE** an accessor method getClientName() with return type string

**DO**

**RETURN** clientName

**END DO**

**CREATE** an accessor method getIssuerBank() with return type string

**DO**

**RETURN** issuerBank

**END DO**

**CREATE** an accessor method getBankAccount() with return type string

**DO**

**RETURN** bankAccount

**END DO**

**CREATE** an accessor method getBalanceAmount() with return type double

**DO**

**RETURN** balanceAmount

**END DO**

**CREATE** an mutator method setClientName (clientName) with return type string

**DO**

**RETURN** clientName

**END DO**

**CREATE** an mutator method setBalanceAmount(balanceAmount) with return type double

**DO**

**RETURN** balanceAmount

**END DO**

**CREATE** the display method

**IF** clientName is empty

**RETURN** "Client name not assigned"

**ELSE**

**RETURN** cardId

**RETURN** clientName

**RETURN** issuerBank

**RETURN** bankAccount

**RETURN** balanceAmount

**IF END**

- **Debit Card**

**CREATE** a DebitCard which is the child class of Bank Card

**DECLARE** instance variable pinNumber as int using private modifier

**DECLARE** instance variable withdrawalAmount as int using private modifier

**DECLARE** instance variable dateOfWithdrawal as string using private modifier

**DECLARE** instance variable hasWithdrawn as boolean using private modifier

**CREATE** a constructor of Debit Card with parameters(cardId, issuerBank, bankAccount ,balanceAmount, clientName , pinNumber)

**CALL** the superclass constructor with parameters( cardId, issuerBank , bankAccount , balanceAmount)

**CALL** the mutator method setClientName(clientName) of superclass

**SET** the parameter value for pinNumber

**SET** the parameter value for hasWithdrawn as false

**SET** the parameter value for withdrawalAmount as 0

**CREATE** an accessor method getPinNumber() with return type int

**DO**

**RETURN** pinNumber

**END DO**

**CREATE** an accessor method getWithdrawalAmount() with return type int

**DO**

**RETURN** withdrawalAmount

**END DO**

**CREATE** an accessor method getDateOfWithdrawal() with return type string

**DO**

**RETURN** dateOfWithdrawal

**END DO**

**CREATE** an accessor method getHasWithdrawn() with return type boolean

**DO**

**RETURN** hasWithdrawn

**END DO**

**CREATE** an mutator method setWithdrawalAmount (withdrawal Amount) with return type int

**DO**

**RETURN** withdrawalAmount

**END DO**

**CREATE** a withdraw method (withdrawalAmount,dateOfWithdrawal , pinNumber)

**IF** pin number is valid

**IF** sufficient amount is present

**SET** the parameter value for withdrawalAmount

**SET** the parameter value for hasWithdrawn as true

**RETURN** "The amount is withdrawn"

**ELSE**

**RETURN** "The PIN number is INVALID!!"

**IF END**

**ELSE**

**RETURN** " You have insufficient balance"

**IF END**

**CREATE** the display method

**CALL** the super class display method

**IF** hasWithdrawn is true

**RETURN** cardId from superclass

**RETURN** clientName from superclass

**RETURN** issuerBank from superclass

**RETURN** bankAccount from superclass

**RETURN** balanceAmount from superclass

**RETURN** pinNumber

**RETURN** withdrawalAmount

**RETURN** dateOfWithdrawal

**ELSE**

**RETURN** balanceAmount from superclass

**IF END**

- **Credit Card**

**CREATE** a CreditCard which is the child class of Bank Card

**DECLARE** instance variable cvcNumber as int using private modifier

**DECLARE** instance variable creditLimit as double using private modifier

**DECLARE** instance variable interestRate as double using private modifier

**DECLARE** instance variable expirationDate as string using private modifier

**DECLARE** instance variable gracePeriod as int using private modifier

**DECLARE** instance variable isGranted as boolean using private modifier

**CREATE** a constructor of Credit Card with parameters(cardId, issuerBank, bankAccount, balanceAmount, clientName, cvcNumber, interestRate, expirationDate)

**CALL** the superclass constructor with parameters( cardId, issuerBank , bankAccount , balanceAmount)

**CALL** the mutator method setClientName(clientName) of superclass

**SET** the parameter value for cvcNumber

**SET** the parameter value for interestRate

**SET** the parameter value for expirationDate

**SET** the parameter value for isGranted as false



**CREATE** an accessor method getCvcNumber() with return type int

**DO**

**RETURN** cvcNumber

**END DO**

**CREATE** an accessor method getCreditLimit() with return type double

**DO**

**RETURN** creditLimit

**END DO**

**CREATE** an accessor method getInterestRate() with return type double

**DO**

**RETURN** interestRate

**END DO**

**CREATE** an accessor method getExpirationDate() with return type string

**DO**

**RETURN** expirationDate

**END DO**

**CREATE** an accessor method getGracePeriod() with return type int

**DO**

**RETURN** gracePeriod

**END DO**

**CREATE** an accessor method getIsGranted() with return type boolean

**DO**

**RETURN** isGranted

**END DO**

**CREATE** a setCreditLimit method (creditLimit , gracePeriod)

**IF** creditLimit is less than or equal to 2.5 times the balance amount

**SET** the parameter value for creditLimit

**SET** the parameter value for gracePeriod

**SET** the parameter value for isGranted as true

**ELSE**

**RETURN** "The credit card cannot be issued . So, credit hasnot been granted to the client"

**IF END**

**CREATE** a cancelCreditCard method

**SET** the parameter value for cvcNumber as 0

**SET** the parameter value for creditLimit as 0

**SET** the parameter value for gracePeriod as 0

**SET** the parameter value for isGranted as false

**RETURN** "Credit Card is cancelled"

**CREATE** the display method

**CALL** the super class display method

**IF** isGranted is true

**RETURN** cvcNumber

**RETURN** creditLimit

**RETURN** interestRate

**RETURN** expirationDate

**RETURN** gracePeriod

**ELSE**

**RETURN** cvcNumber

**RETURN** interestRate

**RETURN** expirationDate

**IF END**

## C. Method Descriptions

- **Bank Card**

The constructor method passes the parameters ( cardId , issuerBank , bankAccount , balanceAmount ) on creation of objects . The accessor method returns the value (CardId, ClientName ,IssuerBank ,BankAccount , BalanceAmount ) of the attributes . The mutator method assigns new value ( newclientName, newbalanceAmount ) to the attribute . The display method displays the output of the Bank Card class.

- **Debit Card**

The constructor method passes the parameters ( cardId , issuerBank ,bankAccount , balanceAmount ,clientName ,pinNumber) on creation of objects .The accessor method returns the value (PinNumber ,WithdrawalAmount ,DateOfWithdrawal,getHasWithdrawn) of the attributes .The mutator method assigns value(withdrawalAmount) to the attribute . The withdraw methods checks if the PIN is correct and set the new balance for sufficient withdraw . The display method displays the output of the Debit Card class.

- **Credit Card**

The constructor method passes the parameters( cardId , issuerBank , bankAccount ,balanceAmount , clientName ,cvcNumber , interestRate , expirationDate) on creation of objects . The accessor method returns the value (CvcNumber,CreditLimit,InterestRate,ExpirationDate,GracePeriod,getIsGranted) of the attributes.The creditlimit method accepts new credit limit ,new grace period and if the credit limit is less than or equal to 2.5 times the balance amount, then the credit is granted.The cancelCreditCard cancels the credit card of the user . The display method displays the output of the Credit Card class .

## D. Testing(Inspection)

- **Test 1 : To inspect the Debit Card class, withdraw the amount, and re-inspect the Debit Card Class**

Test No:	1
Objective:	To inspect the Debit Card class, withdraw the amount, and re-inspect the Debit Card Class
Action:	<ul style="list-style-type: none"><li>➤ The debit card is called with the following arguments: cardId = 5566 issuerBank = "Namaste" bankAccount = "Rojma" balanceAmount = 9900 clientName = "Rojma" pinNumber = 22233</li><li>➤ Inspection of the Debit Card class</li><li>➤ Void withdraw is called with the following arguments: withdrawalAmount = 4500 dateOfWithdrawal = "2023-01-25" pinNumber = 22233</li><li>➤ Re-inspection of the Debit Card class</li></ul>
Expected Result:	The sufficient amount would be withdrawn .
Actual Result:	The amount was withdrawn .
Conclusion:	The test is successful.

Table 1 :To inspect the Debit Card class, withdraw the amount, and re-inspect the Debit Card Class

## Output Result:

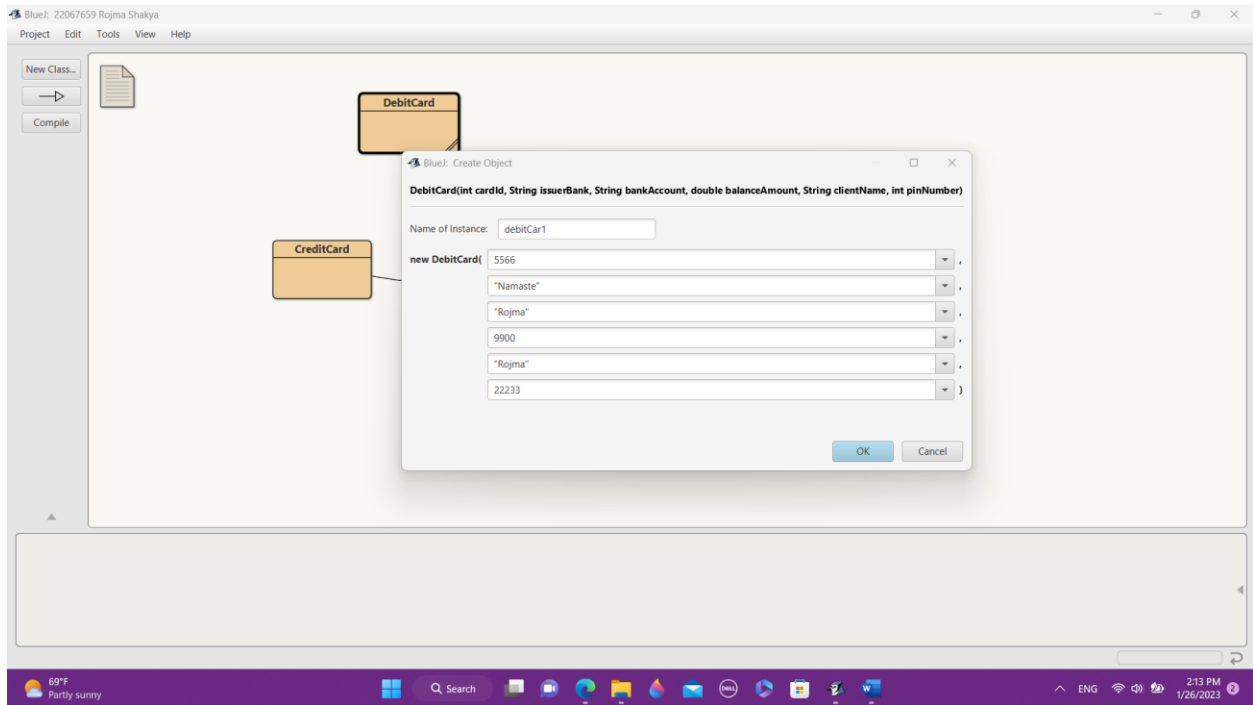


Figure 1: Screenshot of assigning the data in Debit Card class

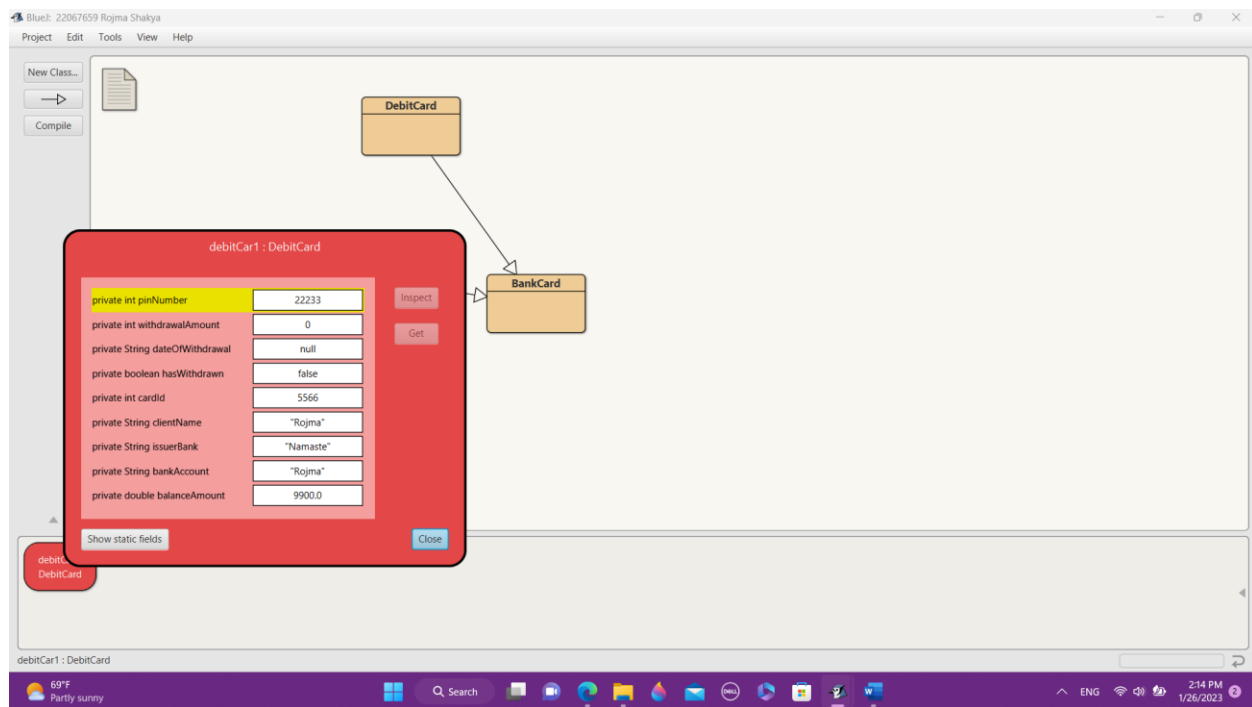


Figure 2: Screenshot for the inspection of Debit Card class

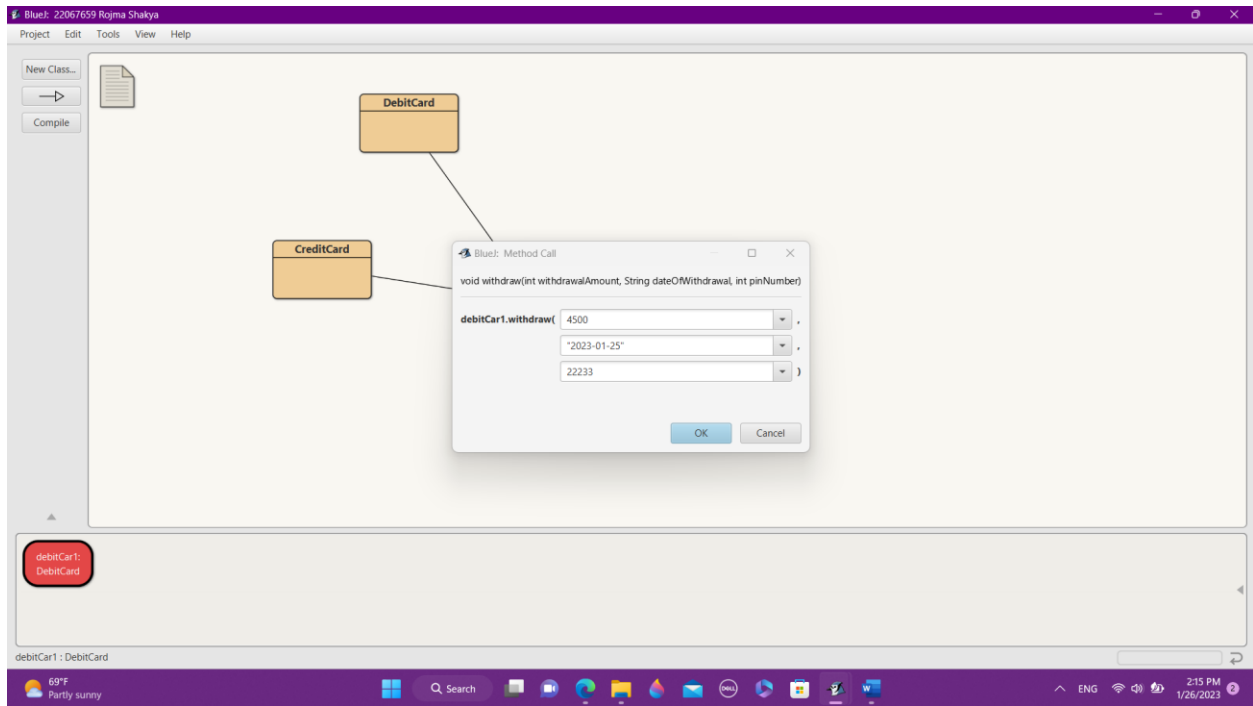


Figure 3: Screenshot of assigning the data in void withdraw in Debit Card class



- **Test 2 : To Inspect Credit Card class, set the credit limit and reinspect the Credit Card class**

<b>Test No:</b>	2
<b>Objective:</b>	To Inspect Credit Card class, set the credit limit and reinspect the Credit Card class
<b>Action:</b>	<ul style="list-style-type: none"> <li>➤ The Credit Card is called with the following arguments:  cardId = 6677  issuerBank = "Nepal"  bankAccount = "Roj"  balanceAmount = 3400  clientName = "Rojma"  cvcNumber = 5590  interestRate = 3400  expirationDate = "2024-01-25"</li> <li>➤ Inspection of the Credit Card class.</li> <li>➤ Void setCreditLimit is called with the following arguments:  creditLimit =889  gracePeriod =600</li> <li>➤ Re-inspection of the credit card class.</li> </ul>
<b>Expected Result:</b>	The credit would be granted to the user.
<b>Actual Result:</b>	The credit was successfully granted to the user.
<b>Conclusion:</b>	The test is successful.

*Table 2: To Inspect Credit Card class, set the credit limit and reinspect the Credit Card class*

## Output Result:

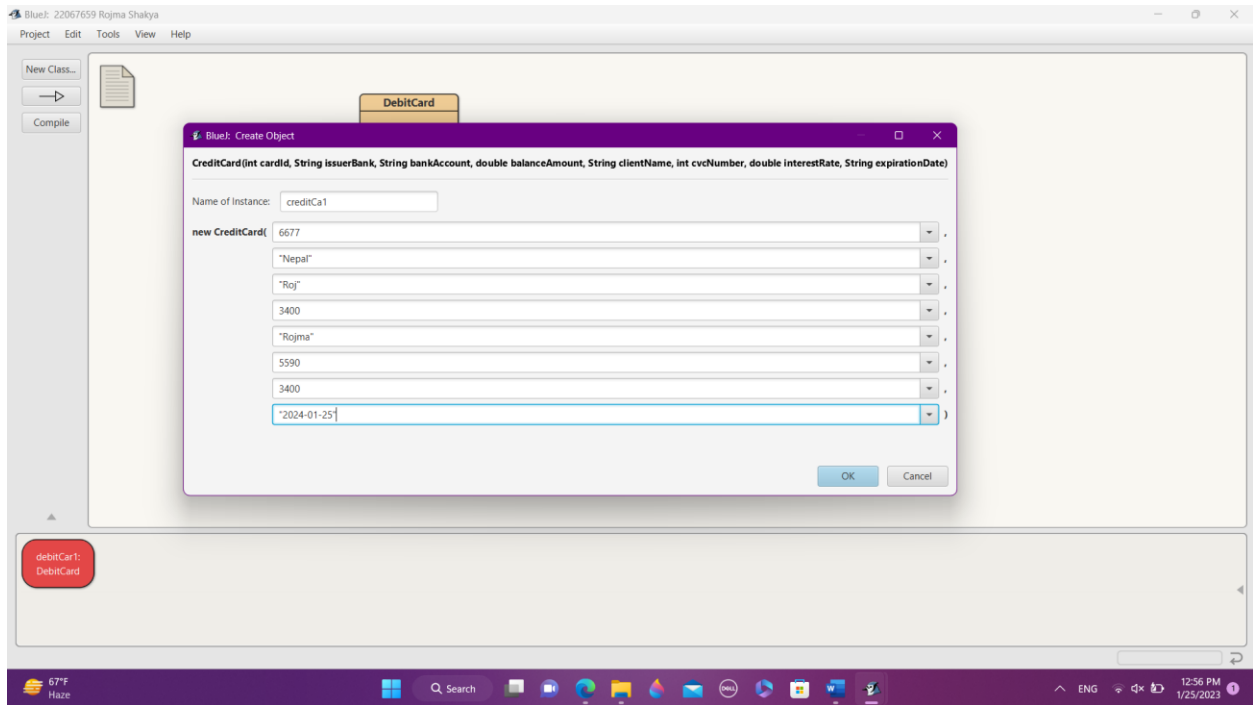


Figure 4: Screenshot of assigning the data in Credit Card class

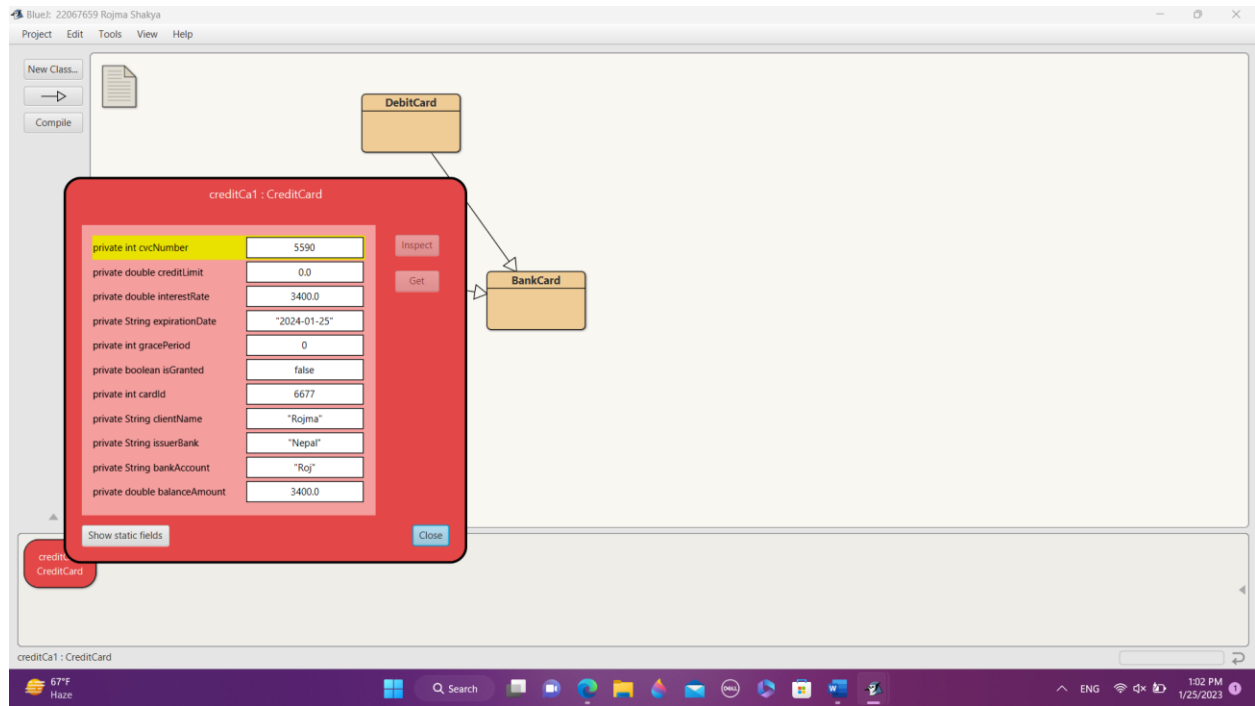


Figure 5: Screenshot for the inspection of Credit Card class

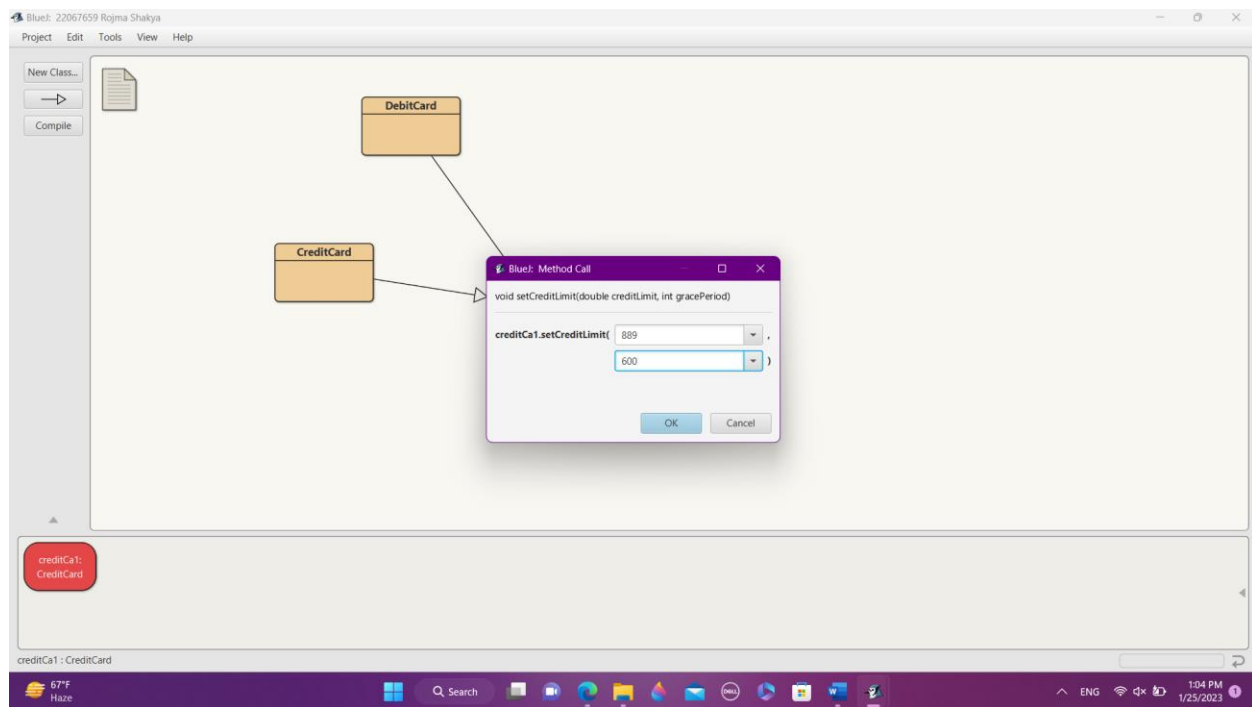


Figure 6: Screenshot of assigning the data in void setcreditLimit in Debit Card class

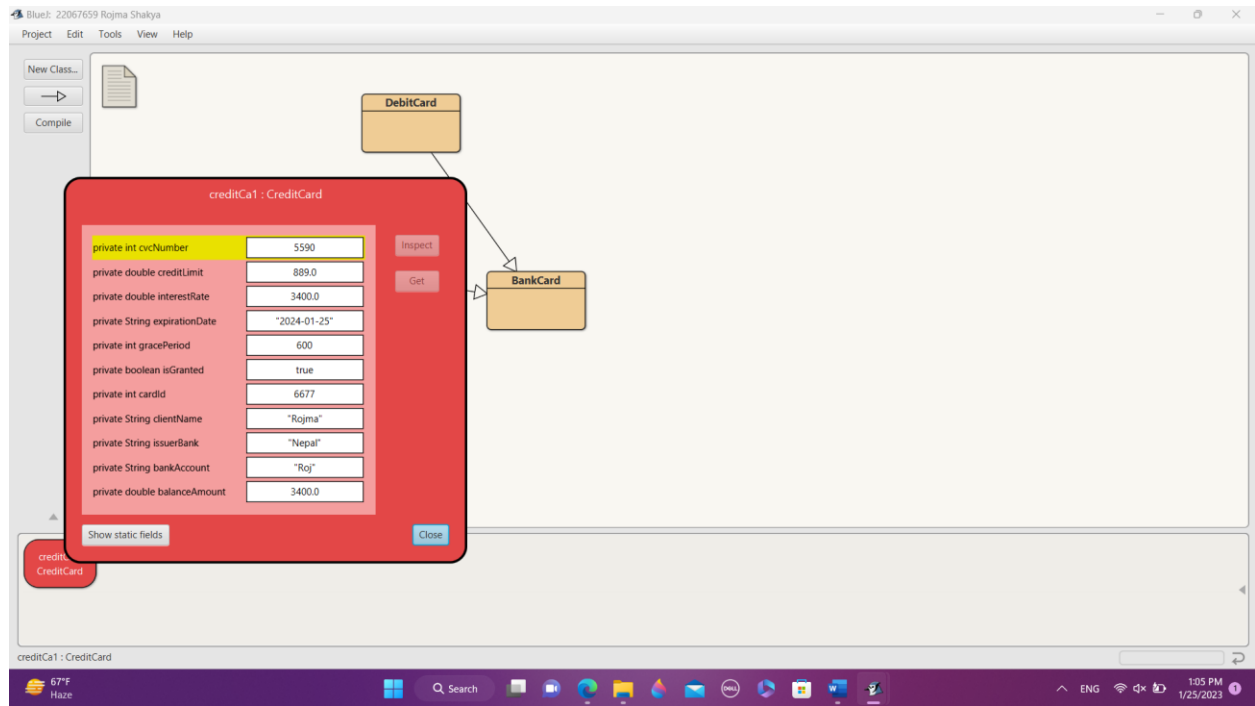


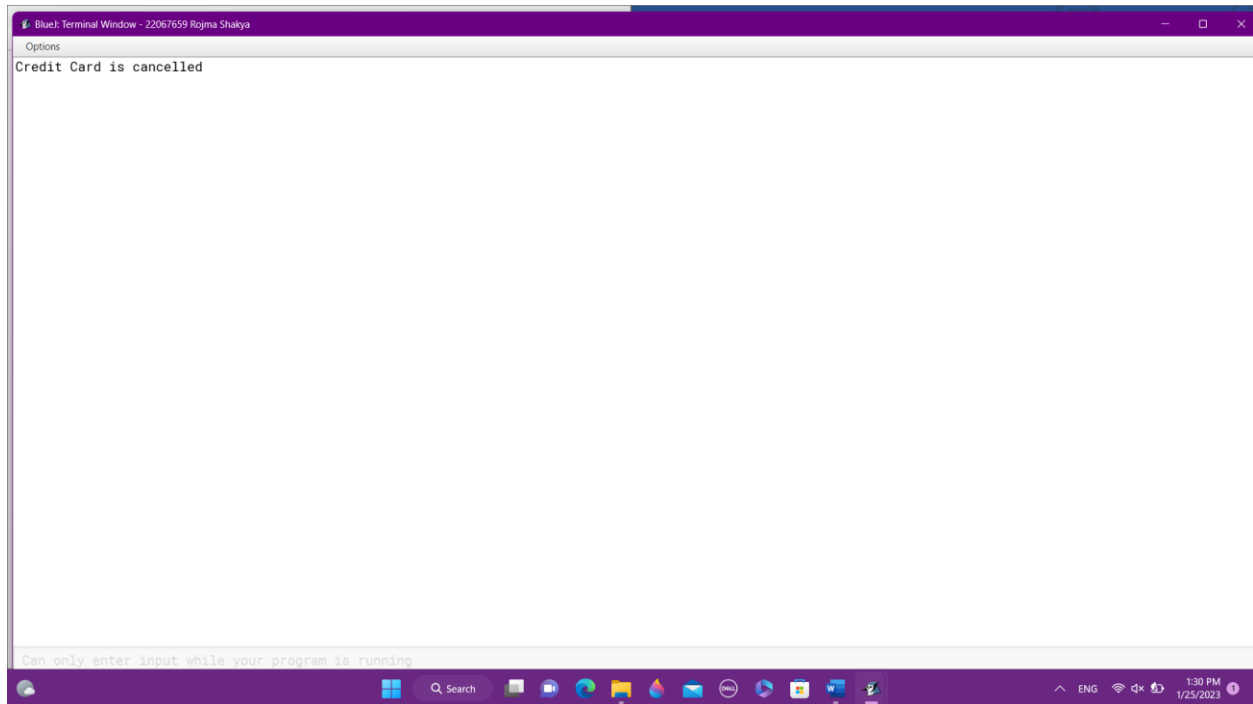
Figure 7: Screenshot for the inspection of void setcreditLimit in Credit Card class

- **Test 3 : To inspect Credit Card class again after cancelling the credit card**

<b>Test No:</b>	3
<b>Objective:</b>	To inspect Credit Card class again after cancelling the credit card
<b>Action:</b>	<ul style="list-style-type: none"> <li>➤ The following arguments after cancelling credit card :  cvcNumber = 0  creditLimit = 0  gracePeriod = 0  isGranted = false</li> </ul>
<b>Expected Result:</b>	The credit would be cancelled for the user.
<b>Actual Result:</b>	The credit was successfully cancelled to the user.
<b>Conclusion:</b>	The test is successful.

*Table 3: To inspect Credit Card class again after cancelling the credit card*

## Output Results:



*Figure 8: Screenshot of the output after credit card is cancelled*

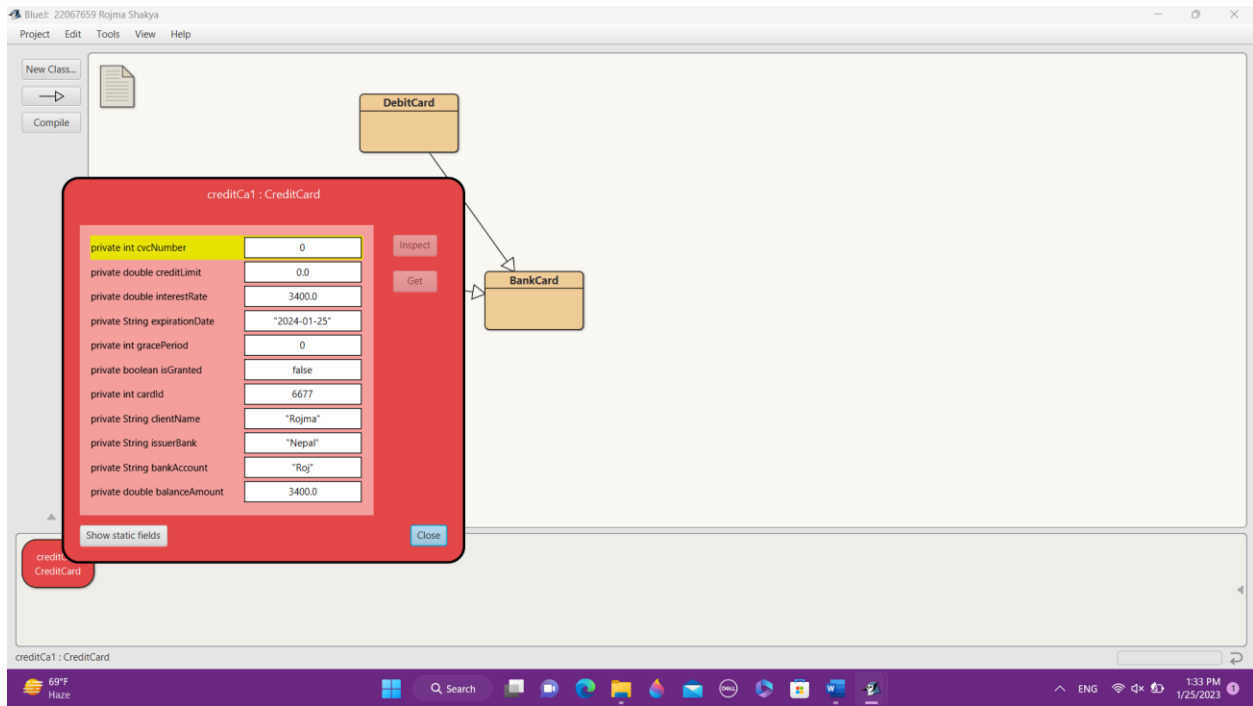


Figure 9: Screenshot for the inspection of cancel Credit Card .



- **Test 4 : To display the details of Debit Card and Credit Card classes**

<b>Test No:</b>	4
Objective:	To display the details of Debit Card and Credit Card classes
Expected Result:	The Debit Card and Credit Card would be displayed.
Actual Result:	The Debit Card and Credit Card is successfully displayed.
Conclusion:	The test is successful.

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

## Output Results:

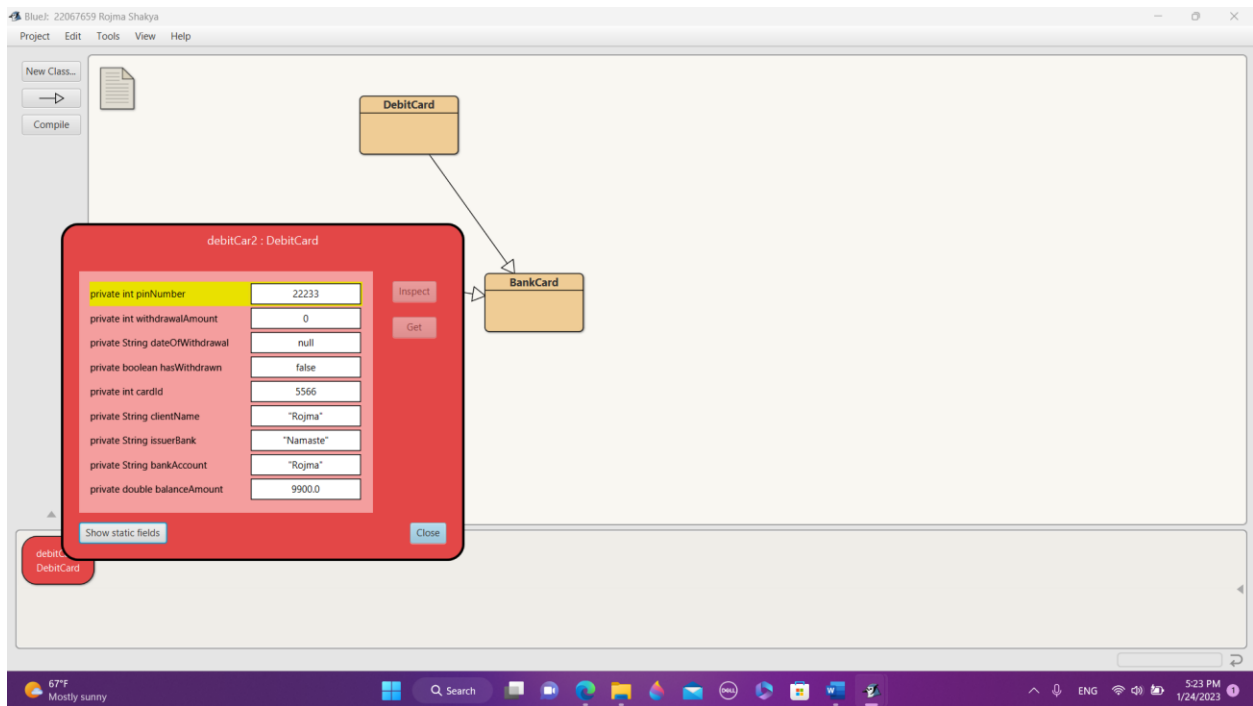


Figure 10: Screenshot of displaying the details of Debit Card.

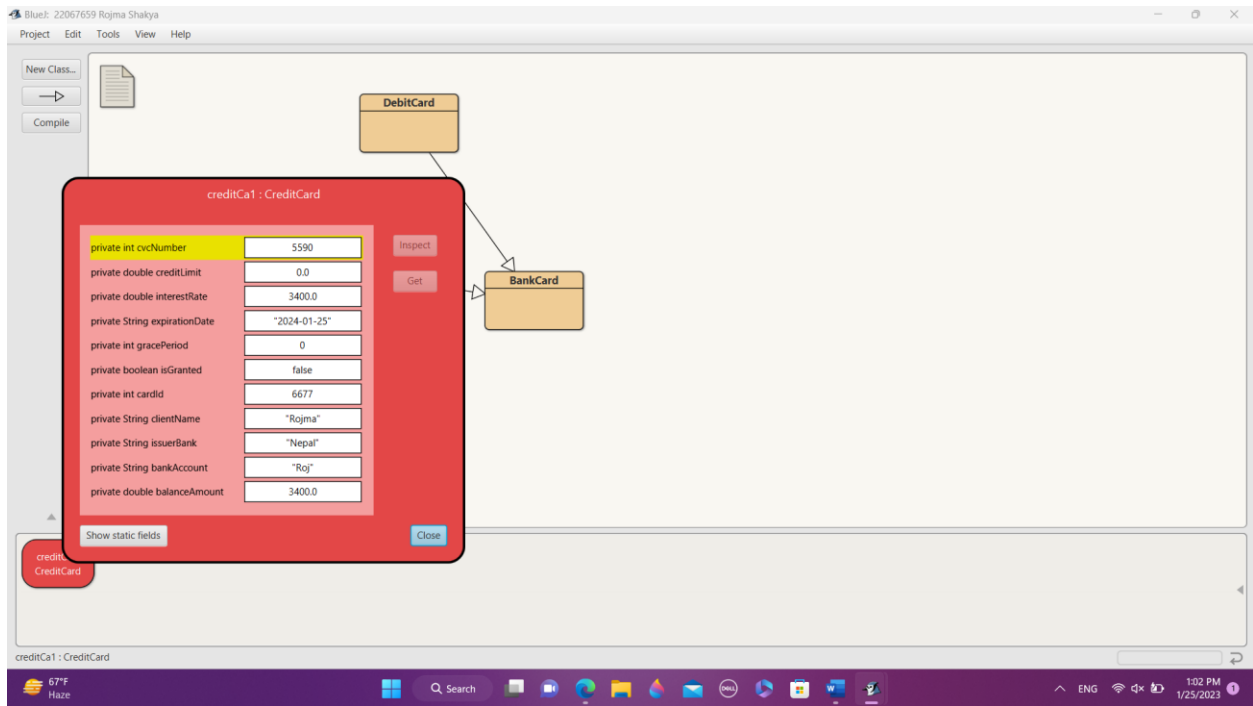


Figure 11: Screenshot of displaying the details of Credit Card.

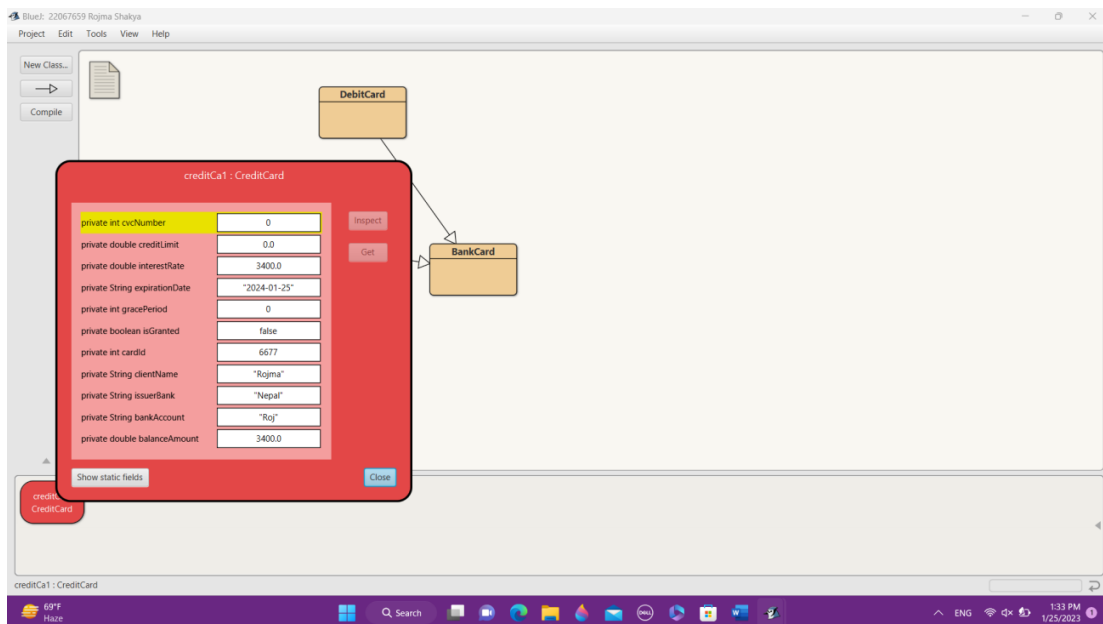


Figure 12: Screenshot of displaying the details of cancel Credit Card

## E. Different error detection and correction

### ○ Syntax errors

Syntax errors is the type of error caused by programmers of misspelled word or of missing punctuation (George, 2022).

## Error detection

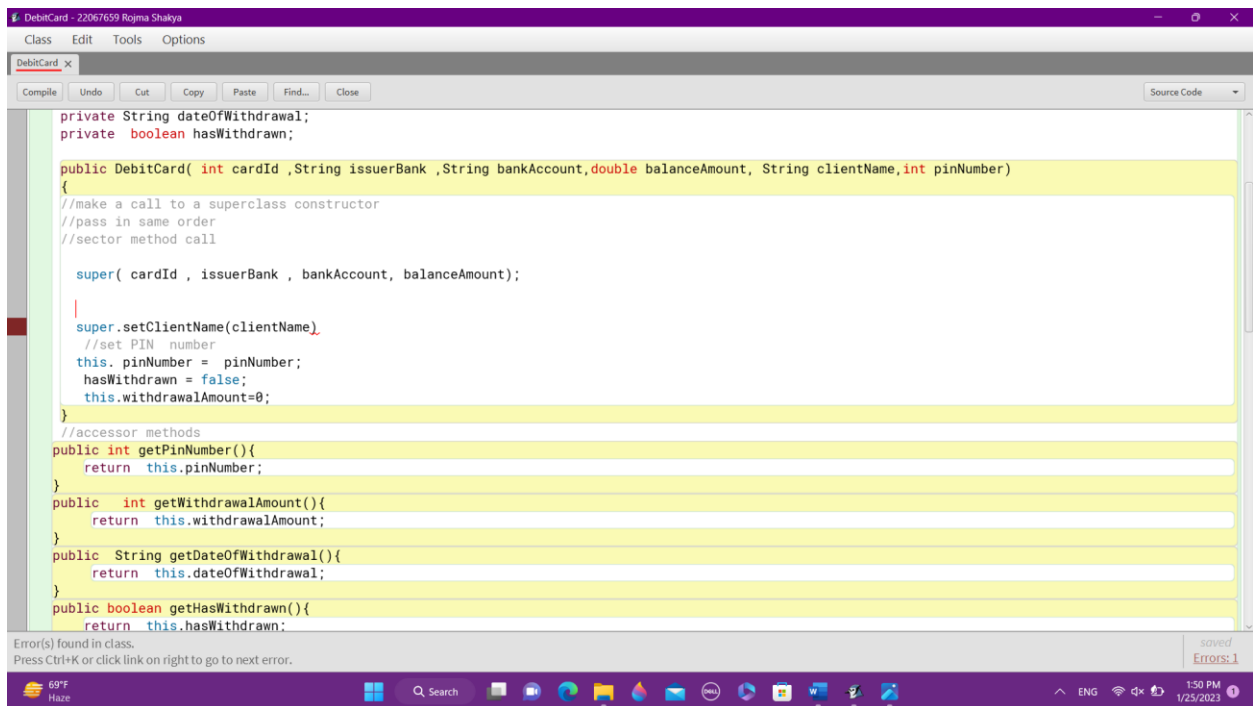
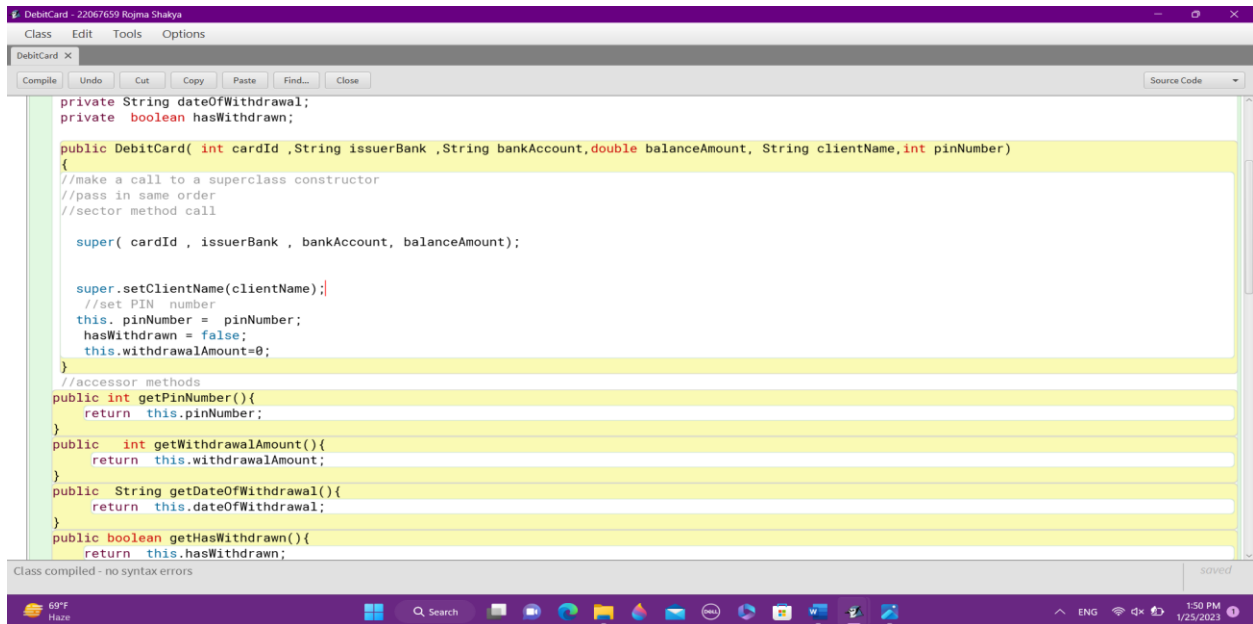


Figure 13: Screenshot of syntax error

## Error correction



The screenshot shows a Java IDE window titled "DebitCard - 22067659 Rogma Shaiya". The menu bar includes "Class", "Edit", "Tools", and "Options". The toolbar has buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The "Source Code" dropdown is set to "Source Code". The code editor displays the following Java code for the DebitCard class:

```
private String dateOfWithdrawal;
private boolean hasWithdrawn;

public DebitCard( int cardId ,String issuerBank ,String bankAccount,double balanceAmount, String clientName,int pinNumber)
{
    //make a call to a superclass constructor
    //pass in same order
    //sector method call

    super( cardId , issuerBank , bankAccount, balanceAmount);

    super.setClientName(clientName);
    //set PIN number
    this.pinNumber = pinNumber;
    hasWithdrawn = false;
    this.withdrawalAmount=0;
}

//accessor methods
public int getPinNumber(){
    return this.pinNumber;
}

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

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

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

At the bottom of the IDE, a status bar indicates "Class compiled - no syntax errors" and "saved". The Windows taskbar at the bottom shows the system clock as 1:50 PM on 1/25/2023, with a temperature of 69°F and weather "Haze".

Figure 14: Screenshot of correction of Syntax error

- **Semantic errors**

Semantic errors are the type of error caused when the code usage isn't correct but the code you have written is shown correct (Mueller, 2016).

## Error Detection

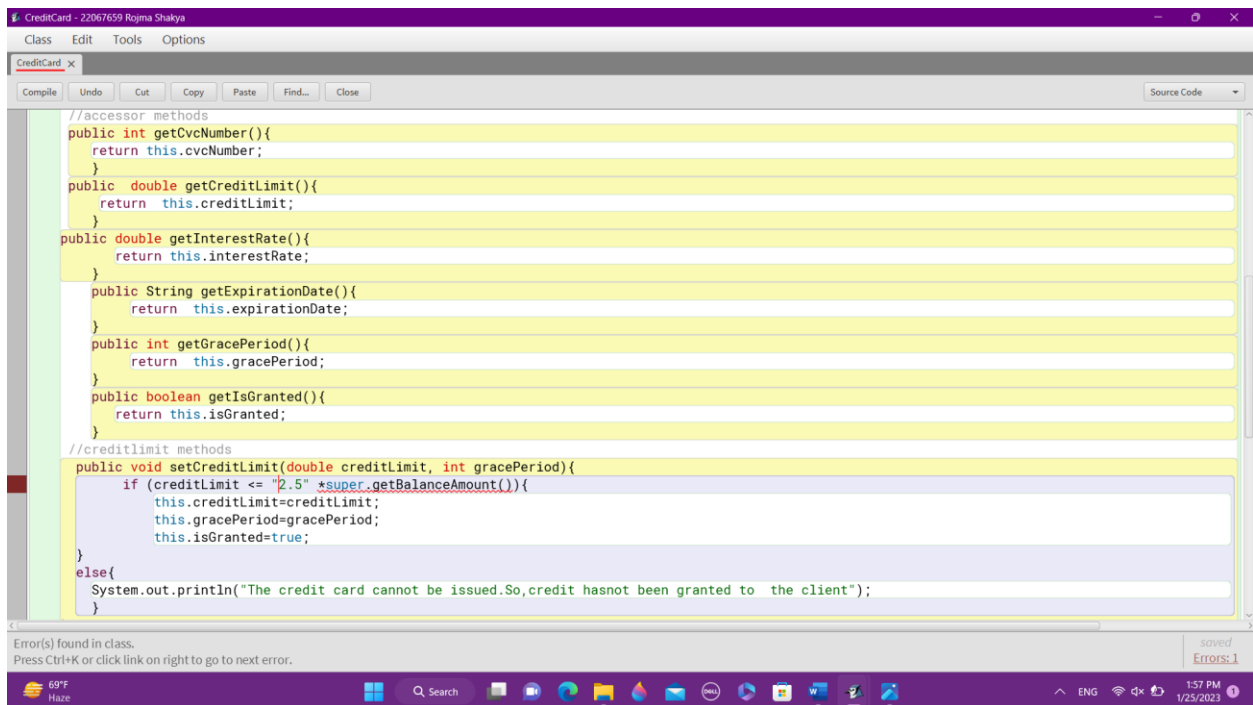
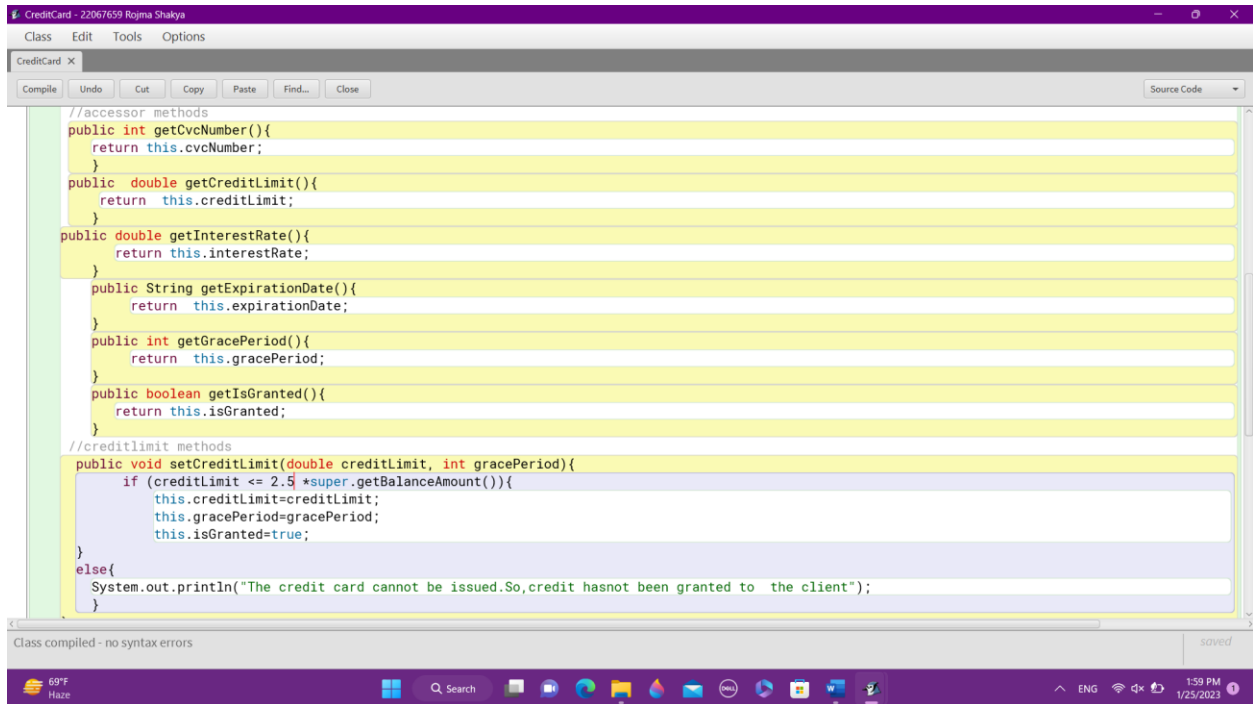


Figure 15: Screenshot of semantic error

## Error Correction



```
// CreditCard - 22067659 Rajna Shakya
Class Edit Tools Options

CreditCard X
Compile Undo Cut Copy Paste Find... Close Source Code

//accessor methods
public int getCvcNumber(){
    return this.cvcNumber;
}
public double getCreditLimit(){
    return this.creditLimit;
}
public double getInterestRate(){
    return this.interestRate;
}
public String getExpirationDate(){
    return this.expirationDate;
}
public int getGracePeriod(){
    return this.gracePeriod;
}
public boolean getIsGranted(){
    return this.isGranted;
}

//creditlimit methods
public void setCreditLimit(double creditLimit, int gracePeriod){
    if (creditLimit <= 2.5 *super.getBalanceAmount()){
        this.creditLimit=creditLimit;
        this.gracePeriod=gracePeriod;
        this.isGranted=true;
    }
    else{
        System.out.println("The credit card cannot be issued.So,credit hasnot been granted to the client");
    }
}

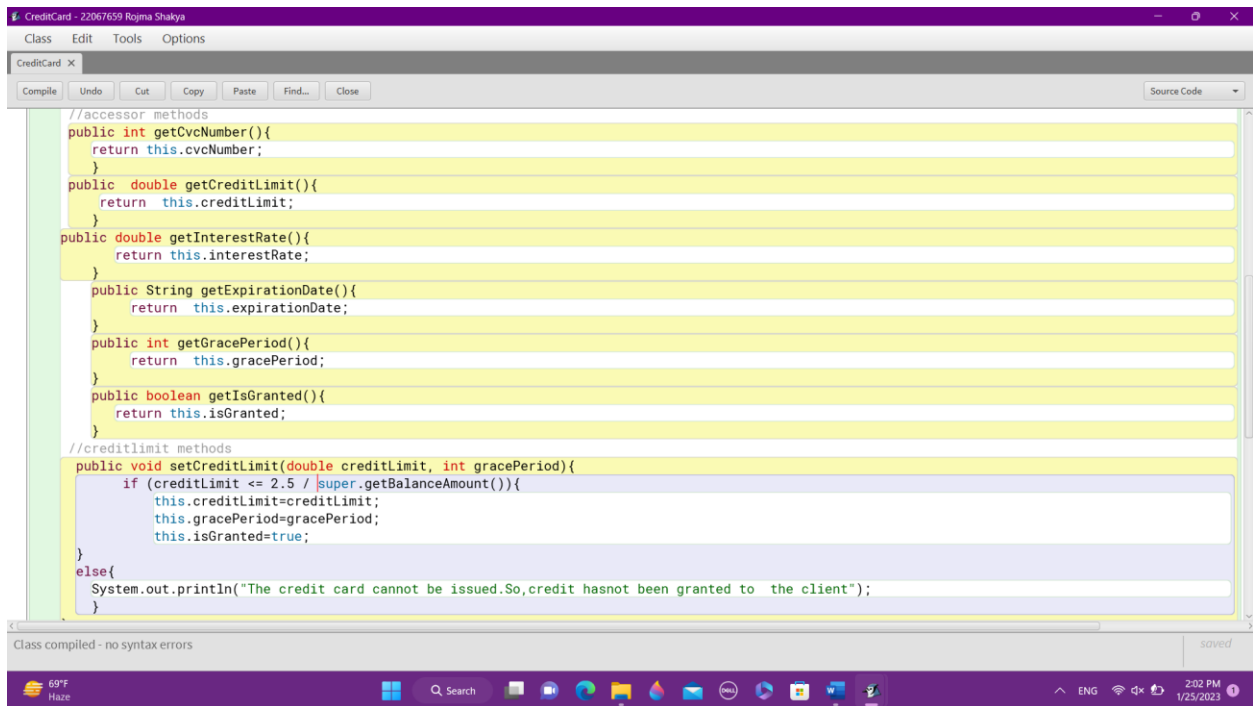
Class compiled - no syntax errors
saved
69°F Haze
Search
ENG 1:59 PM 1/25/2023
```

Figure 16: Screenshot of correction of Semantic error

- **Logical Errors**

Logical Error is the type of error when you think you have written the code perfectly but it hasn't displayed the value like you wanted or isn't performing the task like you wanted (Mueller, 2016).

## Error Detection



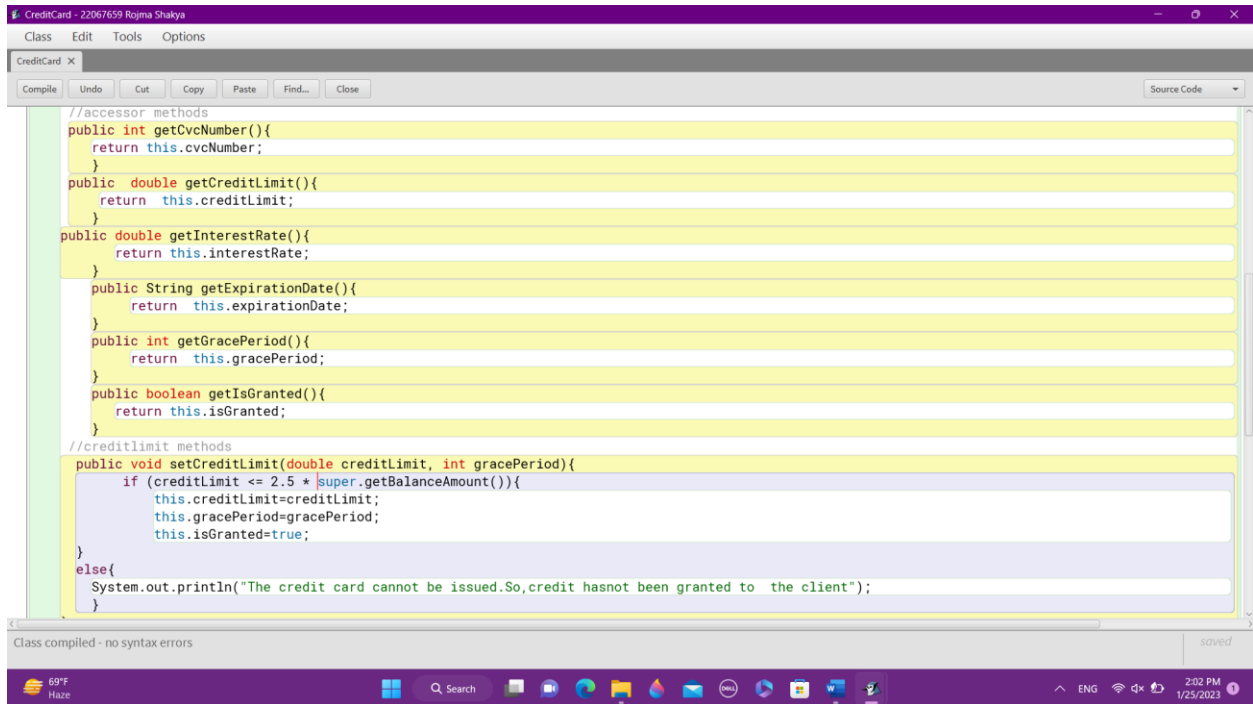
```
//accessor methods
public int getCvcNumber(){
    return this.cvcNumber;
}
public double getCreditLimit(){
    return this.creditLimit;
}
public double getInterestRate(){
    return this.interestRate;
}
public String getExpirationDate(){
    return this.expirationDate;
}
public int getGracePeriod(){
    return this.gracePeriod;
}
public boolean getIsGranted(){
    return this.isGranted;
}

//creditlimit methods
public void setCreditLimit(double creditLimit, int gracePeriod){
    if (creditLimit <= 2.5 / super.getBalanceAmount()){
        this.creditLimit=creditLimit;
        this.gracePeriod=gracePeriod;
        this.isGranted=true;
    }
    else{
        System.out.println("The credit card cannot be issued.So,credit hasnot been granted to the client");
    }
}
```

Figure 17: Screenshot of logical error



## Error Correction



```
//accessor methods
public int getCvcNumber(){
    return this.cvcNumber;
}
public double getCreditLimit(){
    return this.creditLimit;
}
public double getInterestRate(){
    return this.interestRate;
}
public String getExpirationDate(){
    return this.expirationDate;
}
public int getGracePeriod(){
    return this.gracePeriod;
}
public boolean getIsGranted(){
    return this.isGranted;
}

//creditlimit methods
public void setCreditLimit(double creditLimit, int gracePeriod){
    if (creditLimit <= 2.5 * super.getBalanceAmount()){
        this.creditLimit=creditLimit;
        this.gracePeriod=gracePeriod;
        this.isGranted=true;
    }
    else{
        System.out.println("The credit card cannot be issued.So,credit hasnot been granted to the client");
    }
}
```

Class compiled - no syntax errors

Figure 18: Screenshot of correction of logical error

## **F. Conclusion**

In conclusion, from this assignment, I got to learn a lot of things from my coursework, about the coding done in java like the correct use of attributes, the constructors, parameters, when I should use “this”, about accessor and mutator method, difference between parent and child classes and many more which is necessary for the future coders like us. This project was just about how we can make classes in bluej but I have acquired a lot of knowledge. In this assignment, Bank card is a parent class and debit card and credit card are the subclass / child class of Bank card. In which we learned if a valid pin number is entered and sufficient amount is present then the amount can be withdrawn in debit card and in credit card if the credit limit is less than or equal to 2.5 times the balance amount, then the credit is granted to the user. I had a lot difficulties while learning java but due to this coursework I got to realize my difficulties and while I was working on it, I had slowly overcome those difficulties. Otherwise, there were a lot of difficulties like I didn't knew the proper use of inheritance, how I should call the super class in the child class, I had a lot errors but this coursework helped me realized how much basic knowledge I had about java. So, I would like to thank my teachers for helping me during this coursework, during my difficulties because of which I was able to do this coursework on my own. And I would also like to thank my teachers for giving us the opportunities to work on this project because of which I got to learn from my errors, now I am confident that I can do java properly in my future, this helped me realize my confusion of java, which I was confident that I knew but I had a lot of confusion during this project. So, once again I would like to thank my teachers for giving me this coursework and I hope in future I would be getting the coursework which will help me advance more knowledge I have about the java.

## G. References

George, E., 2022. *Cloud Develop.* [Online]  
Available at: <https://clouddevelop.org/syntax-error/>  
[Accessed Wednesday January 2023].

Mueller, J. P., 2016. *dummies.* [Online]  
Available at: <https://www.dummies.com/article/technology/programming-web-design/java/semantic-errors-in-java-153699/>  
[Accessed Wednesday January 2023].

## H. Appendix

- **Bank Card**

```
public class BankCard
{
    //attribute

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

    //constructor

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

        this.clientName="";

        this.cardId=cardId;

        this.issuerBank=issuerBank;

        this.bankAccount=bankAccount;

        this.balanceAmount=balanceAmount;
```

```
}
```

```
//accessor methods
```

```
public int getCardId(){
```

```
    return this.cardId;
```

```
}
```

```
public String getClientName(){
```

```
    return this.clientName;
```

```
}
```

```
public String getIssuerBank(){
```

```
    return this.issuerBank ;
```

```
}
```

```
public String getBankAccount(){
```

```
    return this.bankAccount;
```

```
}
```

```
public double getBalanceAmount(){
```

```
    return this.balanceAmount ;
```

```
}
```

```
//new methods

public void setClientName(String clientName){

    this. clientName=clientName;

}

public void setBalanceAmount(double balanceAmount){

    this. balanceAmount=balanceAmount;

}
```

```
//displaymethod

public void display(){

    if (clientName==""){

        System.out.println("Client name not assigned");

    }else{

        System.out.println("Card ID is"+cardId);

        System.out.println("Client Name is"+ clientName);

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

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

    }

}
```

```

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

    }

}

}

```

- **Debit Card**

```

public class DebitCard extends BankCard
{
    //attribute

    private int pinNumber;

    private int withdrawalAmount;

    private String dateOfWithdrawal;

    private boolean hasWithdrawn;


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

    {

```

```

super( cardId , issuerBank , bankAccount, balanceAmount);

super.setClientName(clientName);

//set PIN number

this. pinNumber = pinNumber;

this.hasWithdrawn = false;

this.withdrawalAmount=0;

}

//accessor methods

public int getPinNumber(){

    return this.pinNumber;

}

public int getWithdrawalAmount(){

    return this.withdrawalAmount;

}

public String getDateOfWithdrawal(){

    return this.dateOfWithdrawal;

}

public boolean getHasWithdrawn(){

```



```

        return this.hasWithdrawn;
    }

    //mutator methods

    public void setWithdrawalAmount(int withdrawalAmount){

        this.withdrawalAmount= withdrawalAmount;

    }

    //Withdraw methods

    public void withdraw(int withdrawalAmount,String dateOfWithdrawal,int pinNumber){

        if(this. pinNumber == pinNumber){

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

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

                this.withdrawalAmount= withdrawalAmount;

                this.hasWithdrawn=true;

                System.out.println("The amount is withdrawn");

            }else{

                System.out.println("The PIN number is INVALID!!");
            }
        }
    }

```

```

    }
}
else{
    System.out.println("You have insufficient balance");
}
}

```

//child display parent display lai call

```

public void display(){
    super.display();
    if( this. hasWithdrawn== true){
        System.out.println("Card ID is"+super.getCardId());
        System.out.println("Client Name is"+ super.getClientName());
        System.out.println("Issuer Bank is"+super.getIssuerBank() );
        System.out.println("Bank Account  is"+ super.getBankAccount());
        System.out.println("Balance  Amount is"+ super.getBalanceAmount());
        System.out.println("PIN number is"+pinNumber);
    }
}

```

```

        System.out.println("Withdrawal Amount is"+ withdrawalAmount);

        System.out.println("Date of withdrawal is"+ dateOfWithdrawal);

    }else{

        System.out.println("Balance Amount is"+ super.getBalanceAmount());

    }

}

}

```

- **Credit Card**

```

public class CreditCard extends BankCard
{
    //attribute

    private int cvcNumber;

    private double creditLimit;

    private double interestRate;

    private String expirationDate;

```

```

private int gracePeriod;

private boolean isGranted;


//Constructor

    public CreditCard (int cardId ,String issuerBank ,String bankAccount,double
balanceAmount,String  clientName,int  cvcNumber,  double  interestRate,String
expirationDate){

        super(cardId , issuerBank , bankAccount, balanceAmount);

        super.setClientName(clientName);

        this.cvcNumber= cvcNumber;

        this.interestRate=interestRate;

        this.expirationDate=expirationDate;

        this.isGranted=false;

    }


//accessor methods

public int getCvcNumber(){

    return this.cvcNumber;

}

public double getCreditLimit(){

    return this.creditLimit;

```

```

    }

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

    public String getExpirationDate(){
        return this.expirationDate;
    }

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

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

    //creditlimit methods

    public void setCreditLimit(double creditLimit, int gracePeriod){
        if (creditLimit <= 2.5 * super.getBalanceAmount()){
            this.creditLimit=creditLimit;
            this.gracePeriod=gracePeriod;
            this.isGranted=true;
        }
    }

    else{

```

```
        System.out.println("The credit card cannot be issued.So,credit hasnot been granted  
to the client");
```

```
    }
```

```
}
```

```
//cancelCreditCard
```

```
public void cancelCreditCard(){
```

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

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

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

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

```
    System.out.println("Credit Card is cancelled");
```

```
}
```

```
//display methods
```

```
public void display(){
```

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

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

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

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

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

```
        System.out.println("Expiration Date is"+expirationDate);

        System.out.println("Grace Period is"+gracePeriod);
    }else{

        System.out.println("CVC number is"+cvcNumber);

        System.out.println("Interest Rate is"+interestRate);

        System.out.println("Expiration Date is"+expirationDate);

    }

}

}
```













