CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Aryan Pradhan

London Met ID: 22068077

College ID: np01cp4a220392

Group: L1C16

Assignment Due Date: Friday, January 27, 2023

Assignment Submission Date: Friday, January 27, 2023

Table of Contents

1)Introduction 4
BlueJ: 4
Microsoft Word: 4
Draw.IO : 4
2) Class Diagram: 5
2.1) Class Diagram of BankCard 5
2.2) Class Diagram of DebitCard 6
2.3) Class Diagram of CreditCard 7
2.4) Class Diagram of Credit Card, Debit Card, Bank Card 8
3) Pseudocode 9
3.1) Pseudocode of bankcard: 9
3.2) Pseudocode of DebitCard: 12
3.3) Pseudocode of Credit Card 15
4) Method Description 19
4.1) Method Description of Bank Card 19
4.2) Method description of DebitCard 20
4.3) Method description of CreditCard 21
5) Tests 22
5.1) Test Case 1: To Inspect the DebitCard 22
Objective 22
To withdraw fund from debit card and reinspect the debitcard 22
Action 22
-The DebitCard is calling the following arguments: 22
Expected Results 22
Definite Results 22
Conclusion 22
5.2) Test Case 2: To inspect CreditCard 25
5.3) Test Case 3: To cancel the creditcard 28
5.4) Test Case 4: To display the details of debit and credit card 29
6) Error Detection and Correction 31
6.1) Syntax Error 31

6.2) Logical Error 33
6.3) Semantic Error 35
7)Conclusion 37
8)Appendix 38
8.1) Appendix of BankCard 38
8.2) Appendix of DebitCard 41
8.3) Appendix of CreditCard 44

1)Introduction

The clean sight of this programming module is to create Bank Card, Debit and Credit Card using Java. As my first coursework for this module there were obvious circumstances where I felt I was running in a wheel. But as I slowly progressed the sight became clearer and soon I completed my project.

As for the tools I used, I used BlueJ and Microsoft Word:

BlueJ:
As an easy to use source for code. All my coding part was covered in Blue J.

Microsoft Word:
As per this report for the coursework , I used Miscrosoft Word, which is a well-known editor for documentation.

Draw.IO :
As for the class diagram part I have used Draw.IO, which appears to be a website. Free and easy to use as all the required sources for this project are available

2) Class Diagram:

Class Diagram shows the classes, properties, operations, and relationships between the objects that make up the structure. For this project the class diagram are as follows:

2.1) Class Diagram of BankCard

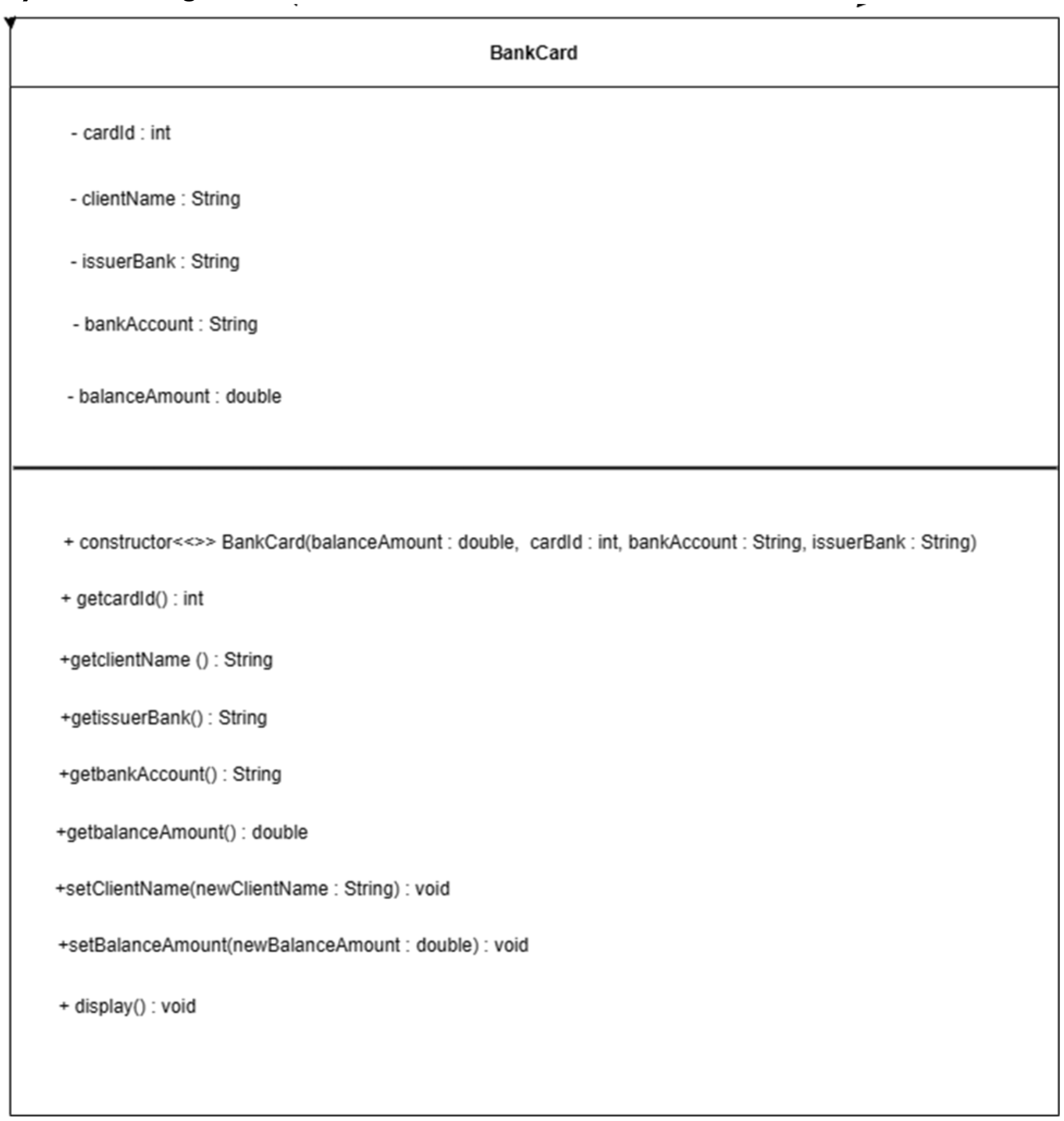
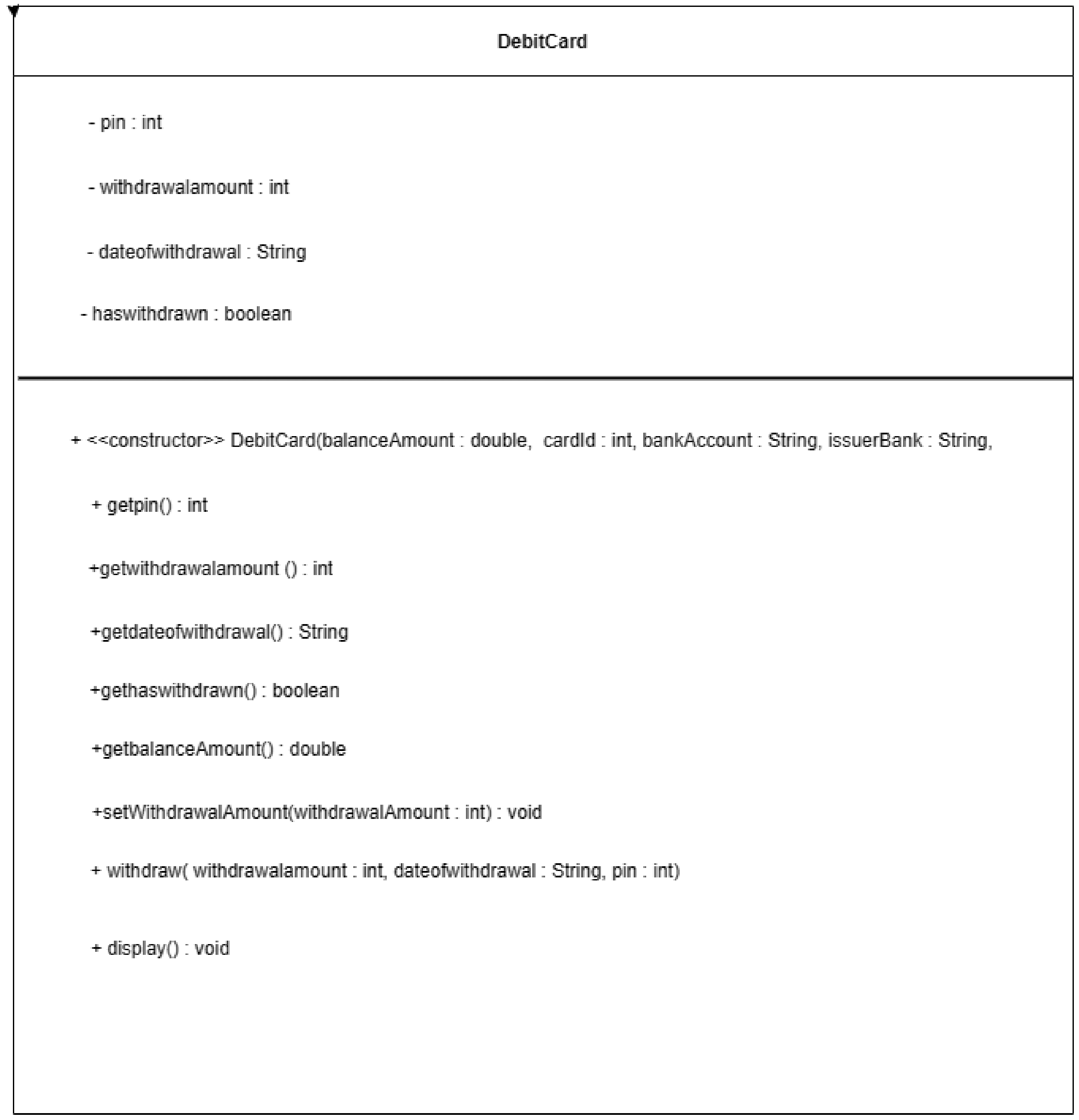

BankCard
- cardId : int
- clientName : String
- issuerBank : String
- bankAccount : String
- balanceAmount : double
+ constructor<<>> BankCard(balanceAmount : double, cardId : int, bankAccount : String, issuerBank : String)
+ getcardId() : int
+getclientName () : String
+getissuerBank() : String
+getbankAccount() : String
+getbalanceAmount() : double
+setClientName(newClientName : String) : void
+setBalanceAmount(newBalanceAmount : double) : void
+ display() : void

Figure 1

2.2) Class Diagram of DebitCard



DebitCard

- pin : int
- withdrawalamount : int
- dateofwithdrawal : String
- haswithdrawn : boolean

+ <<constructor>> DebitCard(balanceAmount : double, cardId : int, bankAccount : String, issuerBank : String,
+ getpin() : int
+getwithdrawalamount () : int
+getdateofwithdrawal() : String
+gethaswithdrawn() : boolean
+getbalanceAmount() : double
+setWithdrawalAmount(withdrawalAmount : int) : void
+ withdraw(withdrawalamount : int, dateofwithdrawal : String, pin : int)
+ display() : void

Figure 2

2.3) Class Diagram of CreditCard

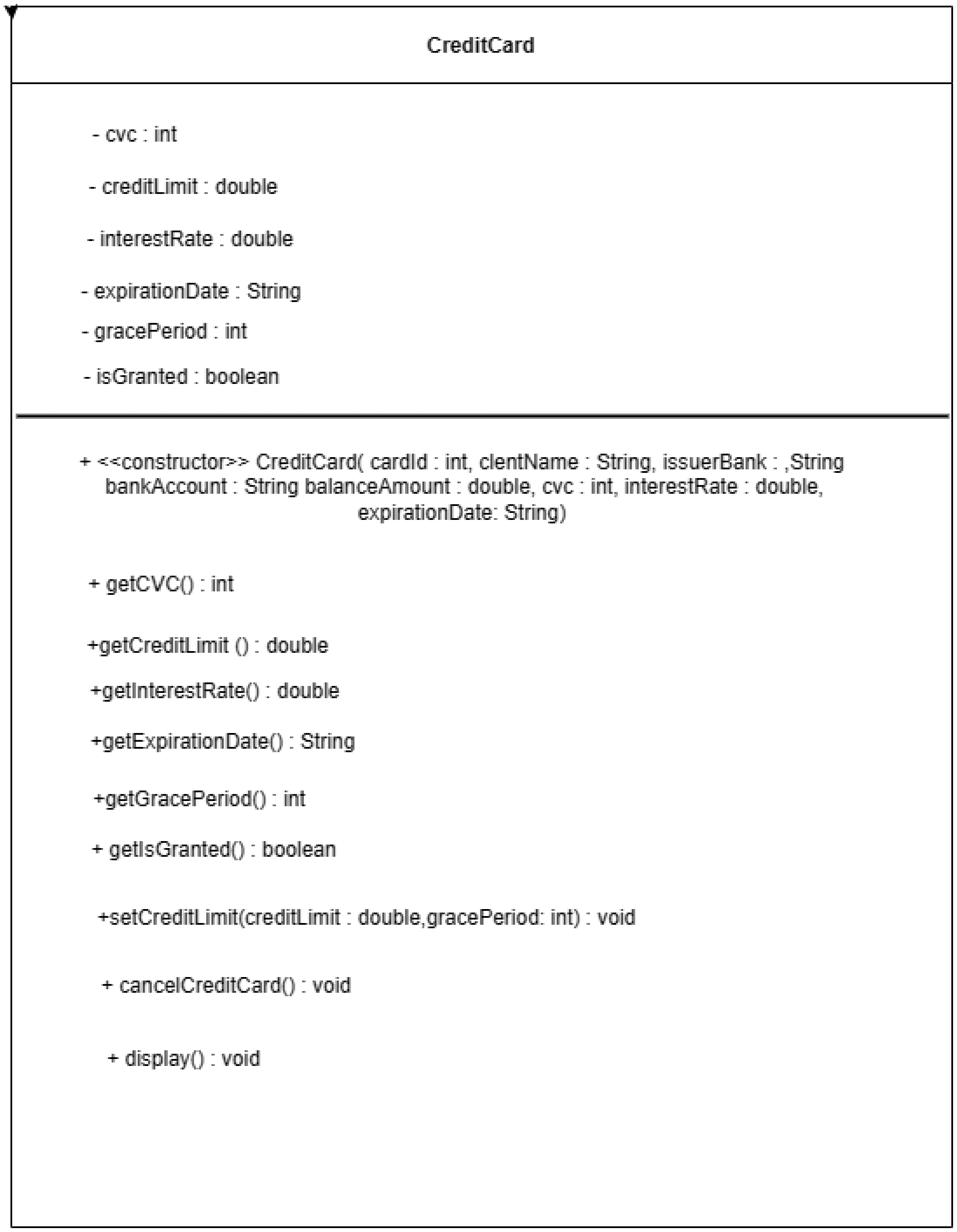

CreditCard
- cvc : int
- creditLimit : double
- interestRate : double
- expirationDate : String
- gracePeriod : int
- isGranted : boolean
+ <<constructor>> CreditCard(cardId : int, clentName : String, issuerBank : ,String bankAccount : String balanceAmount : double, cvc : int, interestRate : double, expirationDate: String)
+ getCVC() : int
+getCreditLimit () : double
+getInterestRate() : double
+getExpirationDate() : String
+getGracePeriod() : int
+ getIsGranted() : boolean
+setCreditLimit(creditLimit : double,gracePeriod: int) : void
+ cancelCreditCard() : void
+ display() : void

Figure 3

2.4) Class Diagram of Credit Card, Debit Card, Bank Card

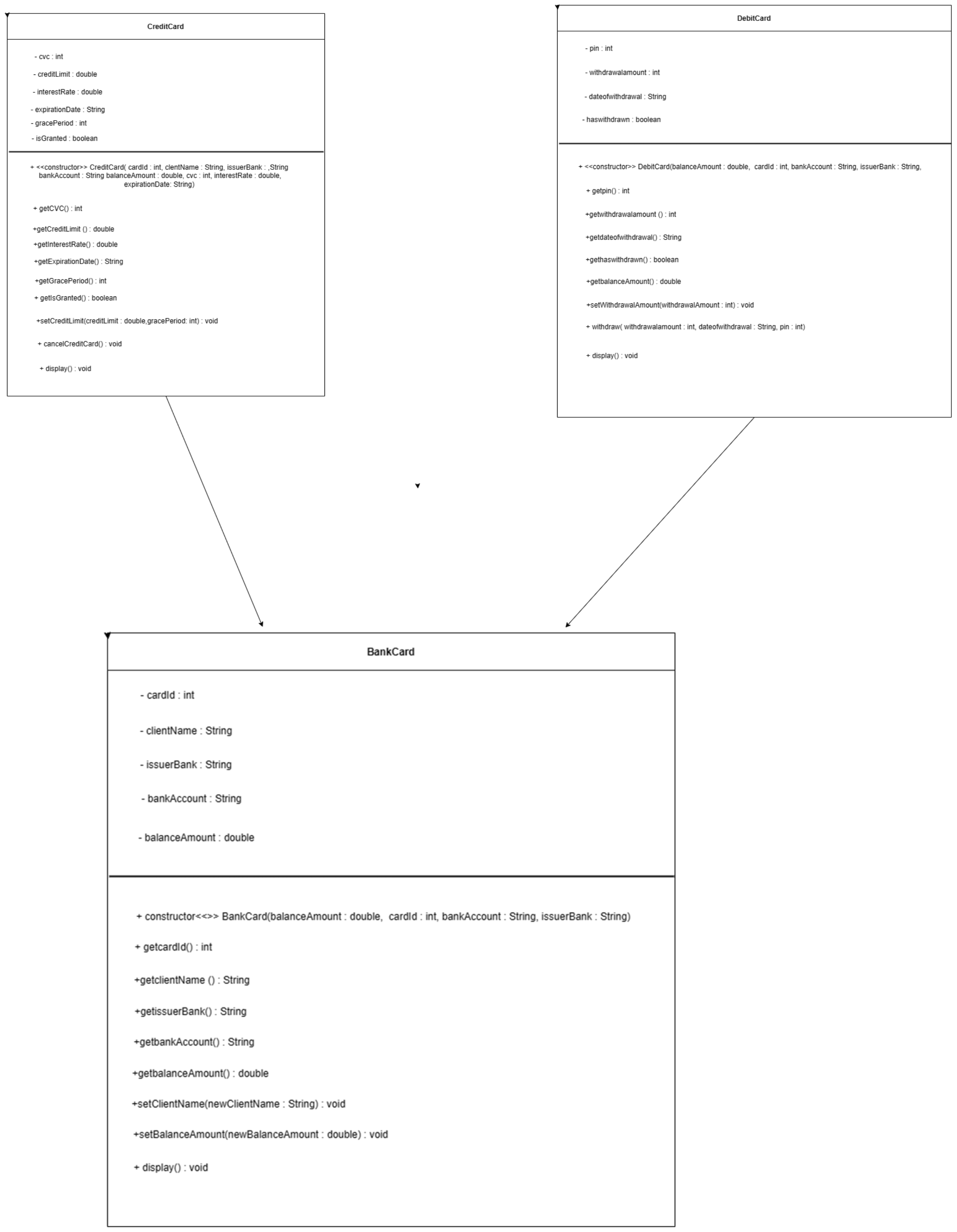

CreditCard
- cvc : int
- creditLimit : double
- interestRate : double
- expirationDate : String
- gracePeriod : int
- isGranted : boolean
+ <<constructor>> CreditCard(cardId : int, clentName : String, issuerBank : ,String bankAccount : String balanceAmount : double, cvc : int, interestRate : double, expirationDate: String)
+ getCVC() : int
+getCreditLimit () : double
+getInterestRate() : double
+getExpirationDate() : String
+getGracePeriod() : int
+ getIsGranted() : boolean
+setCreditLimit(creditLimit : double,gracePeriod: int) : void
+ cancelCreditCard() : void
+ display() : void
DebitCard
- pin : int
- withdrawalamount : int
- dateofwithdrawal : String
- haswithdrawn : boolean
+ <<constructor>> DebitCard(balanceAmount : double, cardId : int, bankAccount : String, issuerBank : String,
+ getpin() : int
+getwithdrawalamount () : int
+getdateofwithdrawal() : String
+gethaswithdrawn() : boolean
+getbalanceAmount() : double
+setWithdrawalAmount(withdrawalAmount : int) : void
+ withdraw(withdrawalamount : int, dateofwithdrawal : String, pin : int)
+ display() : void
BankCard
- cardId : int
- clientName : String
- issuerBank : String
- bankAccount : String
- balanceAmount : double
+ constructor<<>> BankCard(balanceAmount : double, cardId : int, bankAccount : String, issuerBank : String)
+ getcardId() : int
+getclientName () : String
+getissuerBank() : String
+getbankAccount() : String
+getbalanceAmount() : double
+setClientName(newClientName : String) : void
+setBalanceAmount(newBalanceAmount : double) : void
+ display() : void

Figure 4

3) Pseudocode

Pseudocode is algorithm, code written in a simple keyword. Below are the pseudocode for Bank Card, Debit Card, Credit Card

3.1) Pseudocode of bankcard:

```
CREATE class BankCard
DO
    ASSIGNING instance variable cardId as int
    ASSIGNING instance variable clientName as String
    ASSIGNING instance variable issuerBank as String
    ASSIGNING instance variable bankAccount as String
    ASSIGNING instance variable balanceAmount as double

    CREATE constructor BankCard with parameters balanceAmount, cardId,
    bankAccount, issuerBank

    DO
    ASSINGING variable balanceAmount with parameter balanceAmount
    ASSINGING variable cardId with parameter cardId
    ASSINGING variable bankAccount with parameter bankAccount
    ASSINGING variable issuerBank with parameter issuerBank
    ASSINGING variable clientName with parameter clientName
    END DO

    CREATE  accessor method getcardId
    DO
        RETURN cardid
    END DO
```

```
CREATE  accessor method getclientName
DO
        RETURN clientName
END DO
CREATE  accessor method issuerBank
DO
        RETURN issuerBank
END DO
CREATE  accessor method bankAccount
DO
        RETURN bankAccount
END DO
CREATE  accessor method balanceAmount
DO
        RETURN balanceAmount
END DO

CREATE method setClientName with parameter ClientName as String
DO
        RETURN clientName with ClientName
END DO

CREATE method setBalanceAmount with parameter BalanceAmount as double
DO
        RETURN balanceAmount with BalanceAmount
END DO
```

```
    CREATE method to display card details named display
    DO
    IF clientName is Empty
        PRINT Client Name has not been set
    ELSE
        PRINT cardId
        PRINT clientName
        PRINT issuerBank
        PRINT bankAccount
        PRINT balanceAmount
    END DO

END DO
```

3.2) Pseudocode of DebitCard:

```
CREATE class DebitCard which extends BankCard
DO
        ASSINGING variable pin as int
        ASSINGING variable withdrawalamount as int
        ASSINGING variable dateofwithdrawal as String
        ASSINGING variable haswithdrawn as Boolean

        CREATE constructor DebitCard with parameters balanceAmount, cardId,
        bankAccount, issuerBank, clientName, pin

        CALL superclass with balanceAmount, cardId, bankAccount, issuerBank as
        parameters
        CALL superclass ClientName with parameters clientName
        CALL superclass clientName
        ASSINGING variable pin with parameter pin
        ASSINGING variable haswithdrawan as false

        CREATE accessor method getpin
        DO
                RETURN pin
        END DO

        CREATE accessor method getwithdrawalamount
        DO
                RETURN withdrawalamount
        END DO
```

```
CREATE accessor method getdateofwithdrawal
DO
    RETURN dateofwithdrawal
END DO

CREATE accessor method gethaswithdrawn
DO
    RETURN haswithdrawn
END DO

CREATE method setWithdrawalAmount with parameter withdrawalAmount as int
DO
    ASSINGING variable withdrawalamount with parameter withdrawalamount
END DO

CREATE method withdraw with parameters withdrawalamount, dateofwithdrawal
and pin
    DO
    IF pin is equal to pin
        IF balanceAmount is greater than or equal to withdrawalamount
        DO
        SET BalanceAmount with parameters withdrawalamount subtracted
            by getbalanceAmount
        ASSIGN withdrawalamount with parameters withdrawalamount
        ASSIGN dateofwithdrawal with parameters dateofwithdrawal
        ASSIGN haswithdrawan as true
        END DO
        ELSE
            PRINT Insufficient Balance
```

```
                END IF
        ELSE
                PRINT Invalid PIN
        END IF
        END DO

CREATE method display
DO
CALL superclass display
PRINT pin
IF haswithdrawan
        PRINT withdrawalamount
        PRINT dateofwithdrawal
ELSE
        PRINT Transaction not carried out yet.
END DO
```

3.3) Pseudocode of Credit Card

```
CREATE class CreditCard which extends BankCard
DO
    ASSIGNING variable cvc as int
    ASSIGNING variable creditLimit as double
    ASSIGNING variable interestRate as double
    ASSIGNING variable expirationDate as String
    ASSIGNING variable gracePeriod as int
    ASSIGNING variable isGranted as Boolean

    CREATE constructor CreditCard with parameters cardId, clientName,
    issuerBank, bankAccount, cvc, interestRate, expirationDate
    DO
    CALL superclass with parameters balanceAmount, cardId, bankAccount,
            issuerBank
    SET ClientName with parameters clientName
    ASSIGNING variable cvc with parameters cvc
    ASSIGNING variable interestRate with parameters interestRate
    ASSIGNING variable expirationDate with parameters expirationDate
    ASSIGNING variable isGranted as false

    CREATE method getCVC
    DO
        RETURN cvc
    END DO
```

```
CREATE method getCreditLimit
DO
    RETURN creditLimit
END DO

CREATE method getInterestRate
DO
    RETURN interestRate
END DO

CREATE method getExpirationDate
DO
    RETURN expirationDate
END DO

CREATE method getGracePeriod
DO
    RETURN gracePeriod
END DO

CREATE method IsGranted
DO
    RETURN isGranted
END DO
```

```
CREATE method setCreditLimit with parametres creditLimit , gracePeriod
DO
    IF creditLimit is less than or equal to 2.5 times getbalanceAmount
    DO
        ASSIGN creditLimit with parameters creditLimit
        ASSIGN gracePeriod with parameters gracePeriod
        ASSIGN isGranted as true
    END DO
    ELSE
    DO
        PRINT Credit cannot be issued
    END DO
END DO

CREATE method cancelCreditCard
DO
    IF isGranted
    DO
        SET variable cvc, creditLimit, gracePeriod value at 0
        SET isGranted as false
    END DO
    ELSE
    DO
        PRINT Invalid Operation
    END DO
END DO
```

```
CREATE method display
DO
    CALL superclass display
    IF isGranted
    DO
        PRINT cvc
        PRINT creditLimit
        PRINT interestRate
        PRINT expirationDate
        PRINT gracePeriod
    END DO
    ELSE
    DO
        PRINT Credit not granted
    END DO
END DO
```

4) Method Description

4.1) Method Description of Bank Card

1) getcardId(): This is a getter method with an int datatype, it gets cardId and returns it.

2) getclientName(): This is a getter method with an String datatype, it gets clientName and returns it.

3) getissuerBank(): This is a getter method with an String datatype, it gets issuerBank and returns it.

4) getbankAccount(): This is a getter method with an String datatype, it gets bankAccount and returns it.

5) getbalanceAmount(): This is a getter method with a String datatype, it gets balanceAmount and returns it.

6) setClientName(): This is a setter method with parameters newClientName as a String and returns clientName.

7) setBalanceAmount(): This is a setter method with parameters newBalanceAmount as a double and returns balanceAmount.

8) display(): This is a method to display the card details only if clientName is not empty.

4.2) Method description of DebitCard

1) getpin(): This is a getter method with datatype int, it gets and returns pin

2) getwithdrawalamount(): This is a getter method with datatype int, it gets and returns withdrawalamount

3) getdateofwithdrawal(): This is a getter method with datatype String, it gets and returns dateofwithdrawal

4) gethaswithdrawn(): This is a getter method with datatype Boolean, it gets and returns haswithdrawn

5) setWithdrawalAmount(): This is a setter method with parameter as datatype int withdrawalAmount and returns withdrawalAmount\

6) withdraw(): This is a method which allows to withdraw, if pin matches and balanceAmount is greater than or equal to withdrawalamount

7) display(): This method displays the details of the debitcard

4.3) Method description of CreditCard

1) getCVC(): This getter method has datatype int gets and returns cvc

2) getCreditLimit(): This getter method has datatype double gets and returns creditLimit

3) getInterestRate(): This getter method has datatype double gets and returns interestRate

4) getExpirationDate(): This getter method has datatype String gets and returns expirationDate

5) getGracePeriod(): This getter method has datatype int gets and returns gracePeriod

6) getIsGranted(): This getter method has datatype int gets and returns IsGranted

7) setCreditLimit(): This setter method has parameters creditLimit and gravePeriod It checks if credit card can be issued or not. If the creditLimit is Less than or equal to 2.5 times getbalanceAmount then it can be granted orelse not.

8) display(): This setter method displays the credentials of the creditcard.

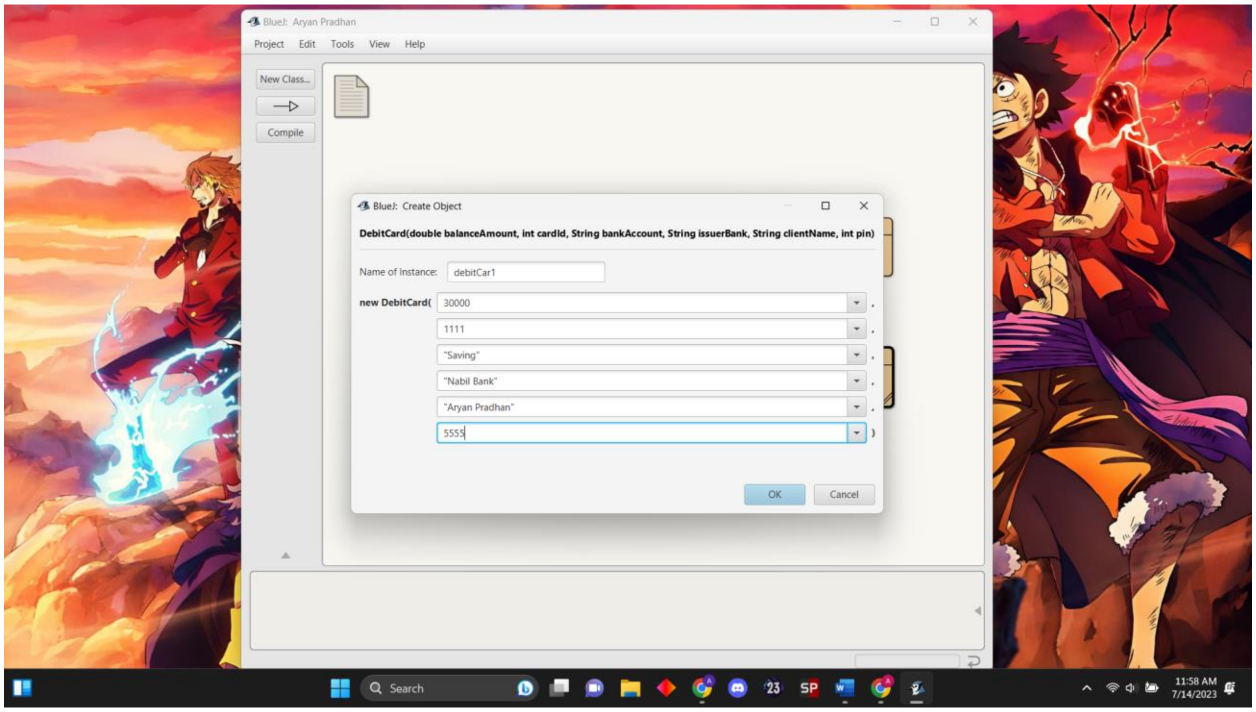
5) Tests

5.1) Test Case 1: To Inspect the DebitCard

TEST	1
Objective	To withdraw fund from debit card and reinspect the debitcard
Action	-The DebitCard is calling the following arguments: cardId = 1111 balanceAmount = 30000 bankAccount = “Saving” clientName = “Aryan Pradhan” issuerBank = “Nabil Bank” pin = 5555 -Inspect DebitCard -void withdraw calls the following arguments -withdrawalamount = 15000 -dateofwithdrawal = 2023-07-14 -pin = 5555 -Reinspect DebitCard
Expected Results	The fund will be withdrawn
Definite Results	The fund has been withdrawn
Conclusion	This test was a success

Table 1

Definite Results



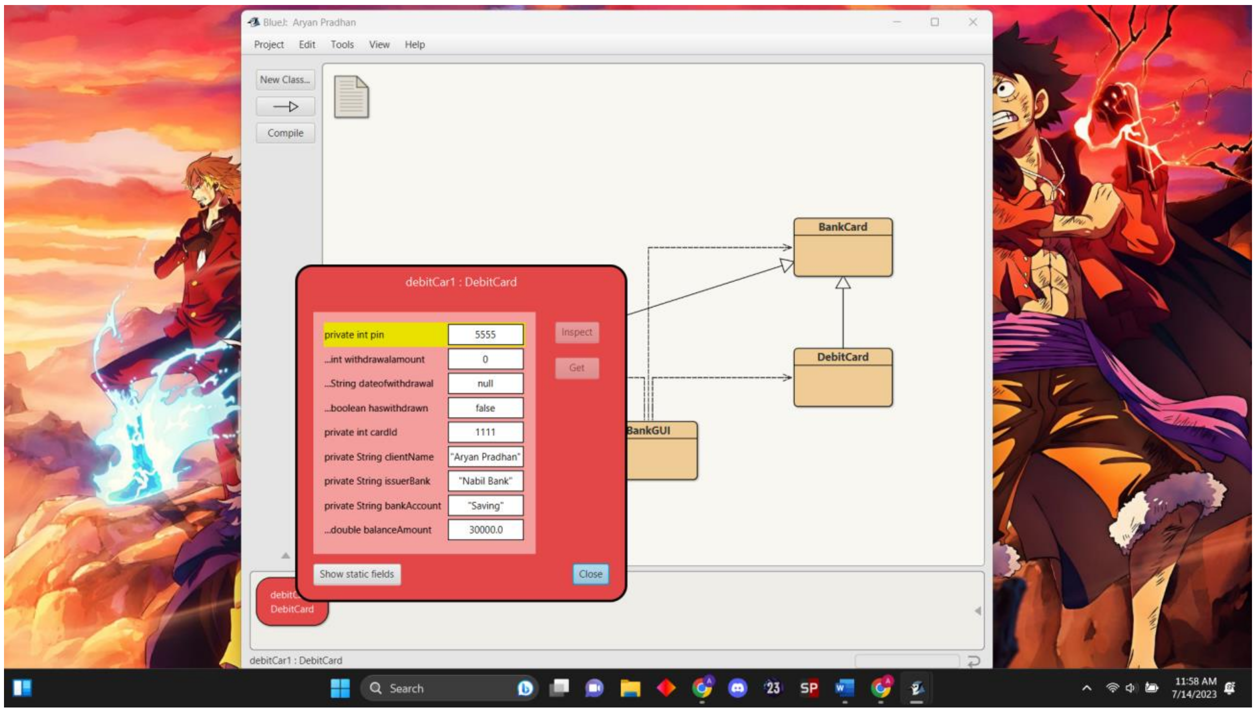
BlueJ: Create Object

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

Name of Instance: debitCar1

new DebitCard(30000, 1111, "Saving", "Nabil Bank", "Aryan Pradhan", 5555)

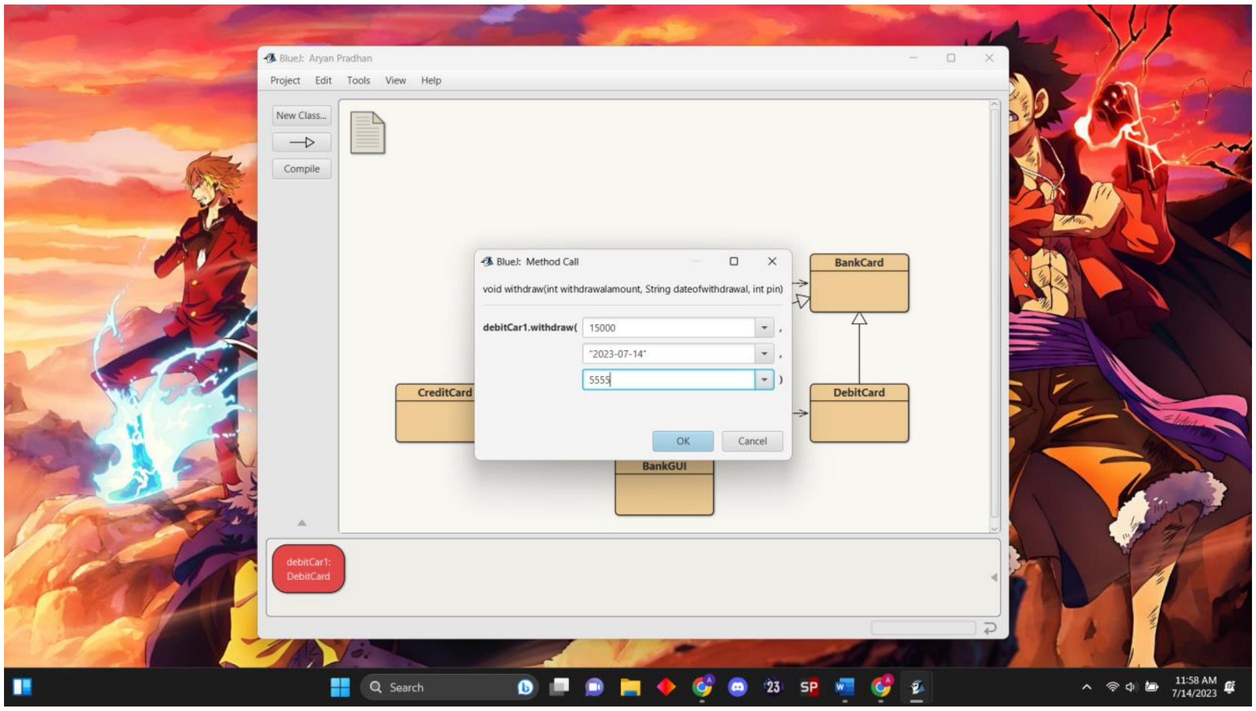
Figure 5



debitCar1 : DebitCard

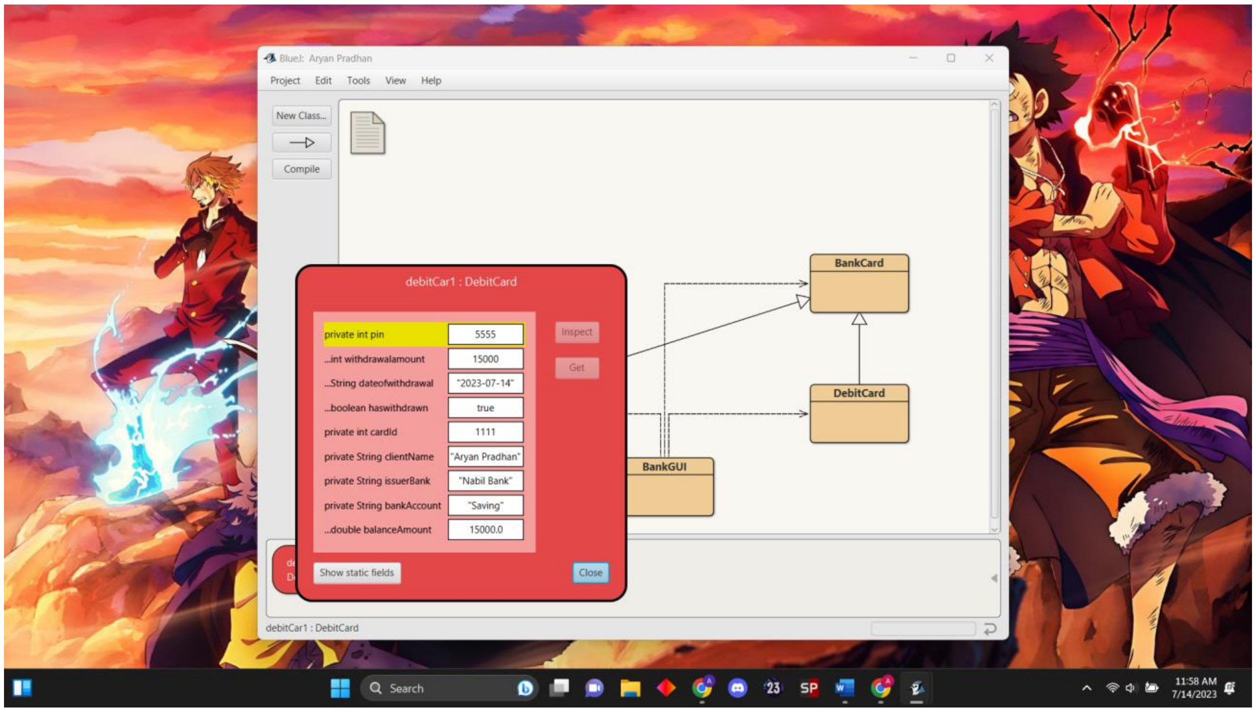
private int pin	5555
...int withdrawalamount	0
...String dateofwithdrawal	null
...boolean haswithdrawn	false
private int cardId	1111
private String clientName	"Aryan Pradhan"
private String issuerBank	"Nabil Bank"
private String bankAccount	"Saving"
...double balanceAmount	30000.0

Figure 6



```
void withdraw(int withdrawalamount, String dateofwithdrawal, int pin)

debitCar1.withdraw( 15000 ,
                    "2023-07-14" ,
                    5555 )
```

Figure 7



private int pin	5555
...int withdrawalamount	15000
...String dateofwithdrawal	"2023-07-14"
...boolean haswithdrawn	true
private int cardId	1111
private String clientName	"Aryan Pradhan"
private String issuerBank	"Nabil Bank"
private String bankAccount	"Saving"
...double balanceAmount	15000.0

Figure 8

5.2) Test Case 2: To inspect CreditCard

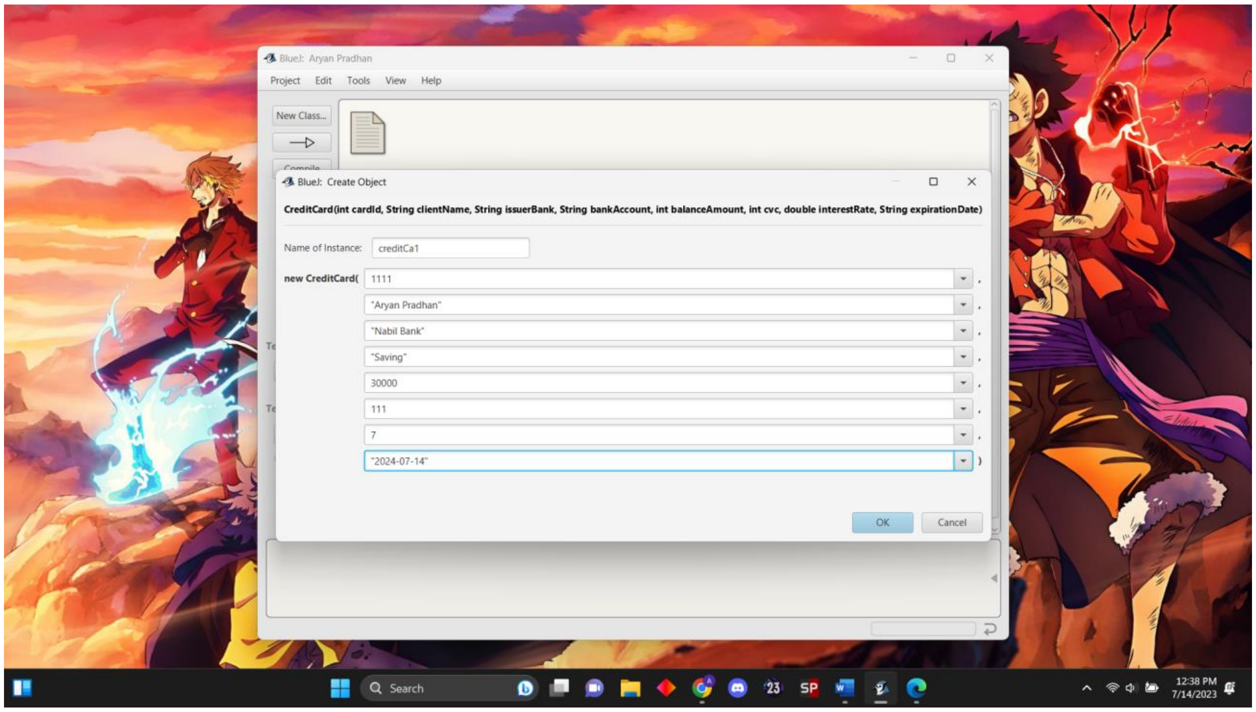
Test	2
Objective	To set the credit limit and reinspect credit card

Action	-The credit card calls the following arguments
	
	balanceAmount = 30000
	cardId = 1111
	bankAccount = “Saving”
	issuerBank = “Nabil Bank”
	cvc = 111
	interestRate = 7
	expirationDate = “2024-07-14”
	
	-void setCreditLimit is calling the following arguments
	
	newcreditLimit = 600
	gracePeriod = 7
	-Reinspect the creditcard

Expected Results	The Credit Limit is set
Definite Results	The Credit Limit has been set
Conclusion	This test was successful

Table 2

Definite Results

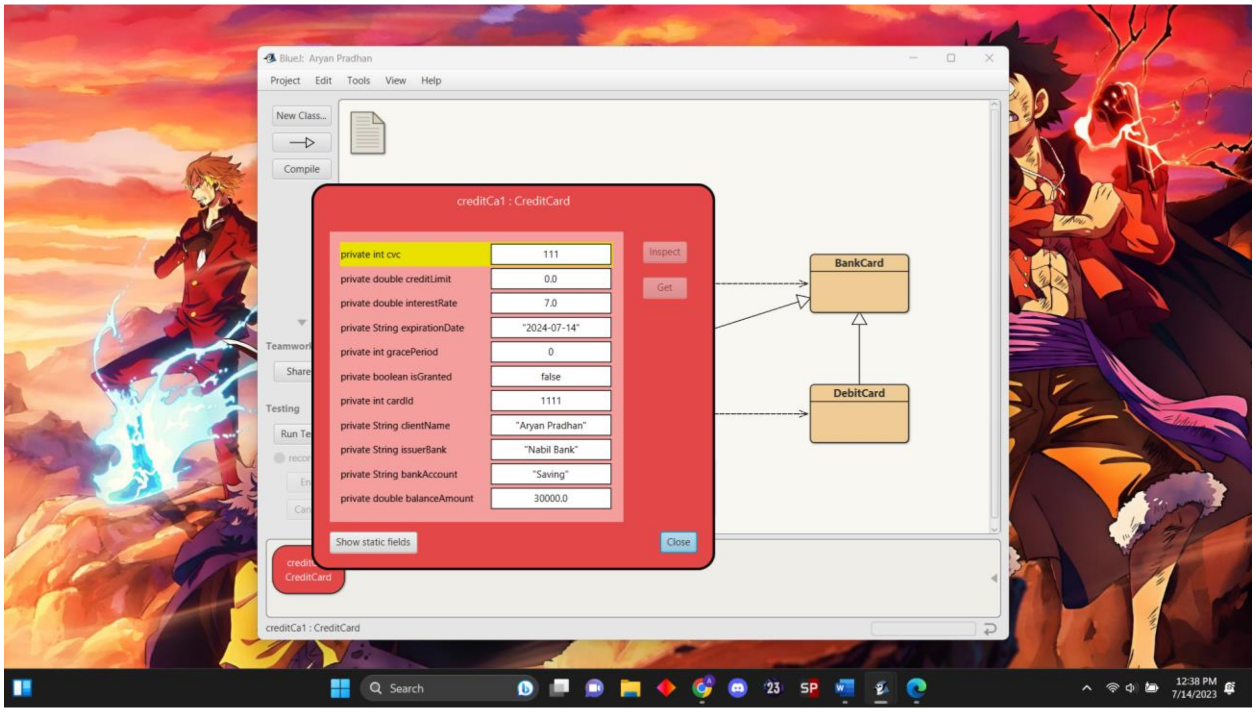


CreditCard(int cardId, String clientName, String issuerBank, String bankAccount, int balanceAmount, int cvc, double interestRate, String expirationDate)

Name of Instance: creditCa1

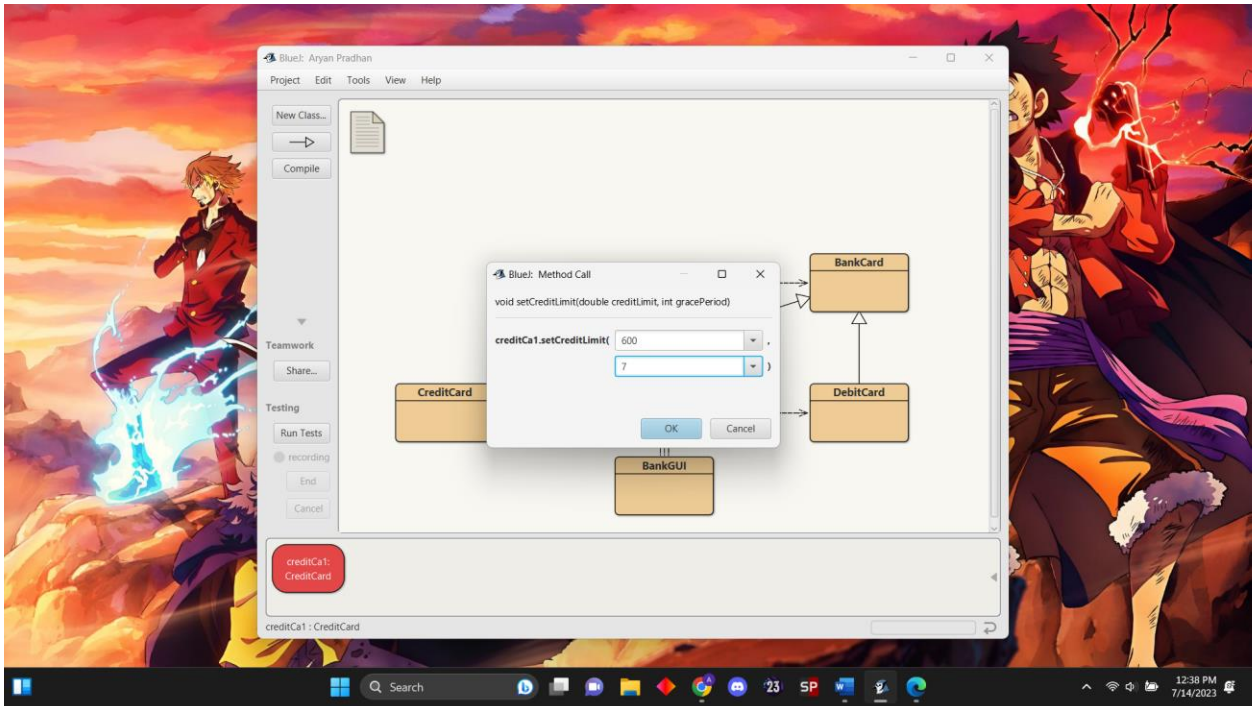
new CreditCard(1111, "Aryan Pradhan", "Nabil Bank", "Saving", 30000, 111, 7, "2024-07-14")

Figure 9



private int cvc	111
private double creditLimit	0.0
private double interestRate	7.0
private String expirationDate	"2024-07-14"
private int gracePeriod	0
private boolean isGranted	false
private int cardId	1111
private String clientName	"Aryan Pradhan"
private String issuerBank	"Nabil Bank"
private String bankAccount	"Saving"
private double balanceAmount	30000.0

Figure 10

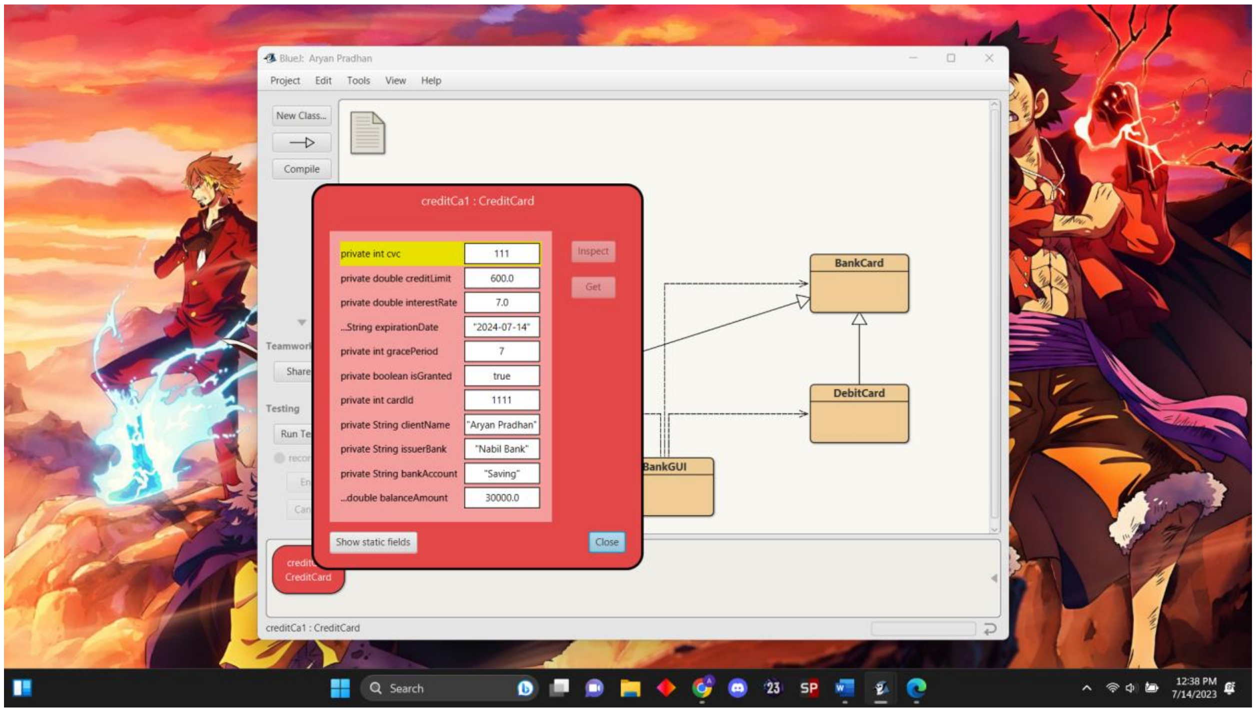


BlueJ: Method Call

void setCreditLimit(double creditLimit, int gracePeriod)

creditCa1.setCreditLimit(600 , 7)

Figure 11



creditCa1 : CreditCard

private int cvc	111
private double creditLimit	600.0
private double interestRate	7.0
...String expirationDate	"2024-07-14"
private int gracePeriod	7
private boolean isGranted	true
private int cardId	1111
private String clientName	"Aryan Pradhan"
private String issuerBank	"Nabil Bank"
private String bankAccount	"Saving"
...double balanceAmount	30000.0

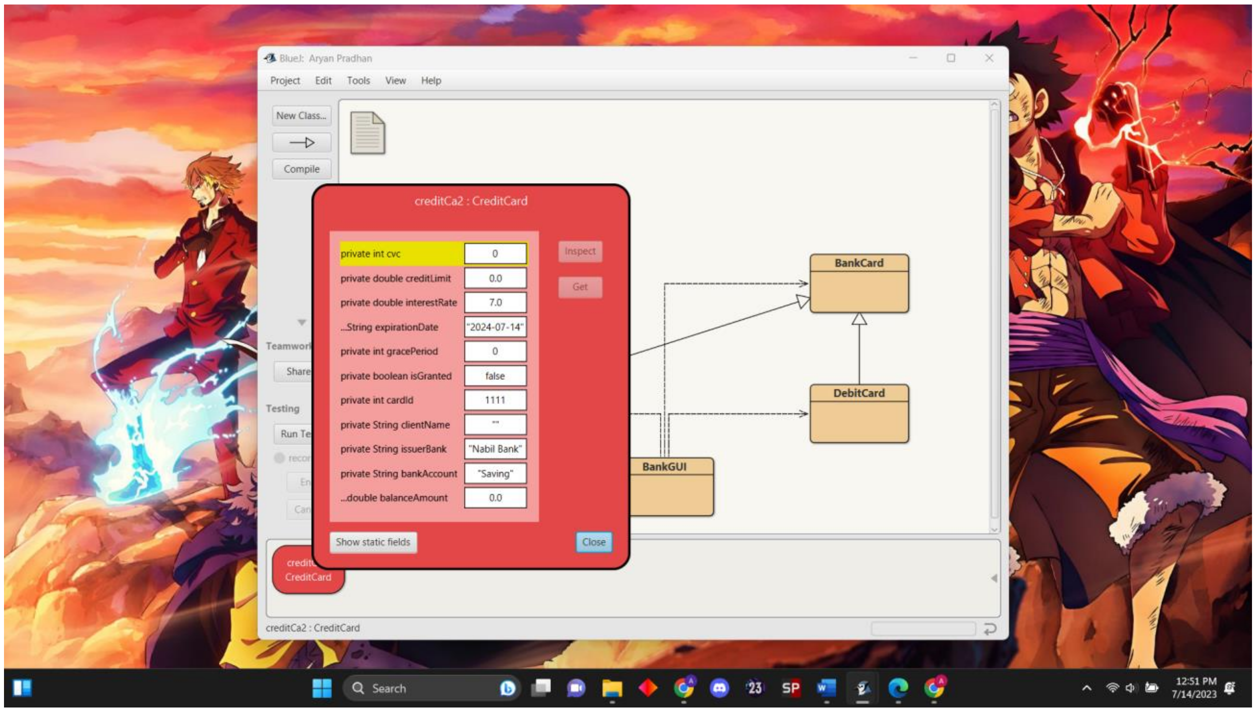
Figure 12

5.3) Test Case 3: To cancel the creditcard

Test	3
Objective	To cancel the creditcard and reinspect
Action	-To cancel the creditcard -To reinspect the creditcard
Expected Result	The card is cancelled
Definite Result	The card has been cancelled
Conclusion	This test was successful

Table 3

<u>Definite Results</u>



creditCa2 : CreditCard

private int cvc	0
private double creditLimit	0.0
private double interestRate	7.0
...String expirationDate	"2024-07-14"
private int gracePeriod	0
private boolean isGranted	false
private int cardId	1111
private String clientName	""
private String issuerBank	"Nabil Bank"
private String bankAccount	"Saving"
...double balanceAmount	0.0

creditCa2 : CreditCard

Figure 13

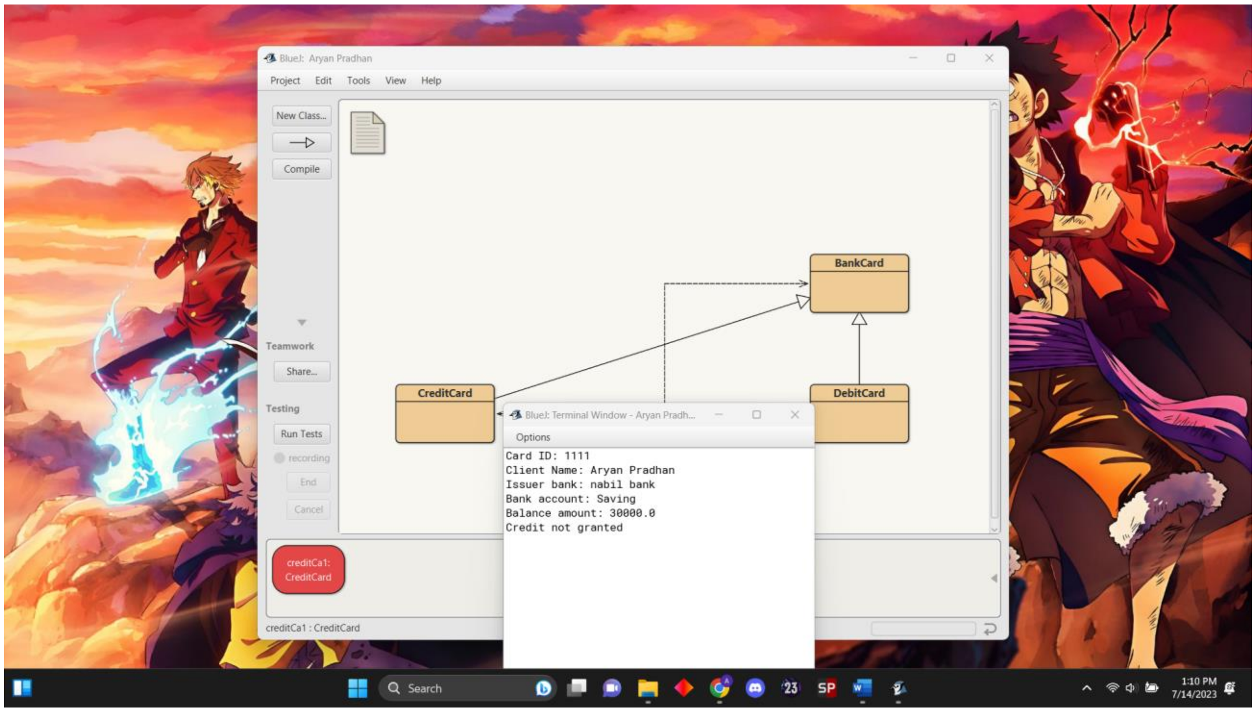
5.4) Test Case 4: To display the details of debit and credit card

Test	4
Objective	To display debit card and credit card
Action	-running the void method of both the cards that'll display the card details
Expected Results	The details of debit card and credit card are displayed
Definite Results	The details of debit card and credit card have been displayed
Conclusion	This test was a success

Table 4

Definite Results

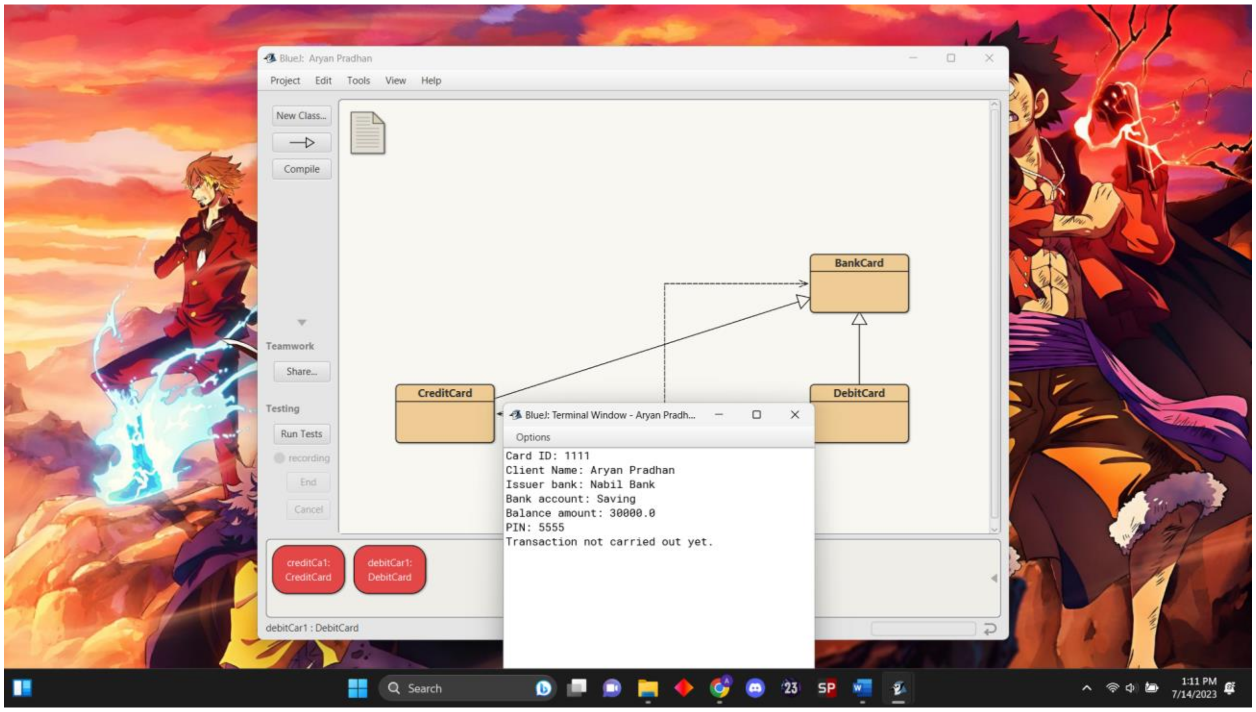
Display of Credit Card



```
Card ID: 1111
Client Name: Aryan Pradhan
Issuer bank: nabil bank
Bank account: Saving
Balance amount: 30000.0
Credit not granted
```

Figure 14

Display of Debit Card



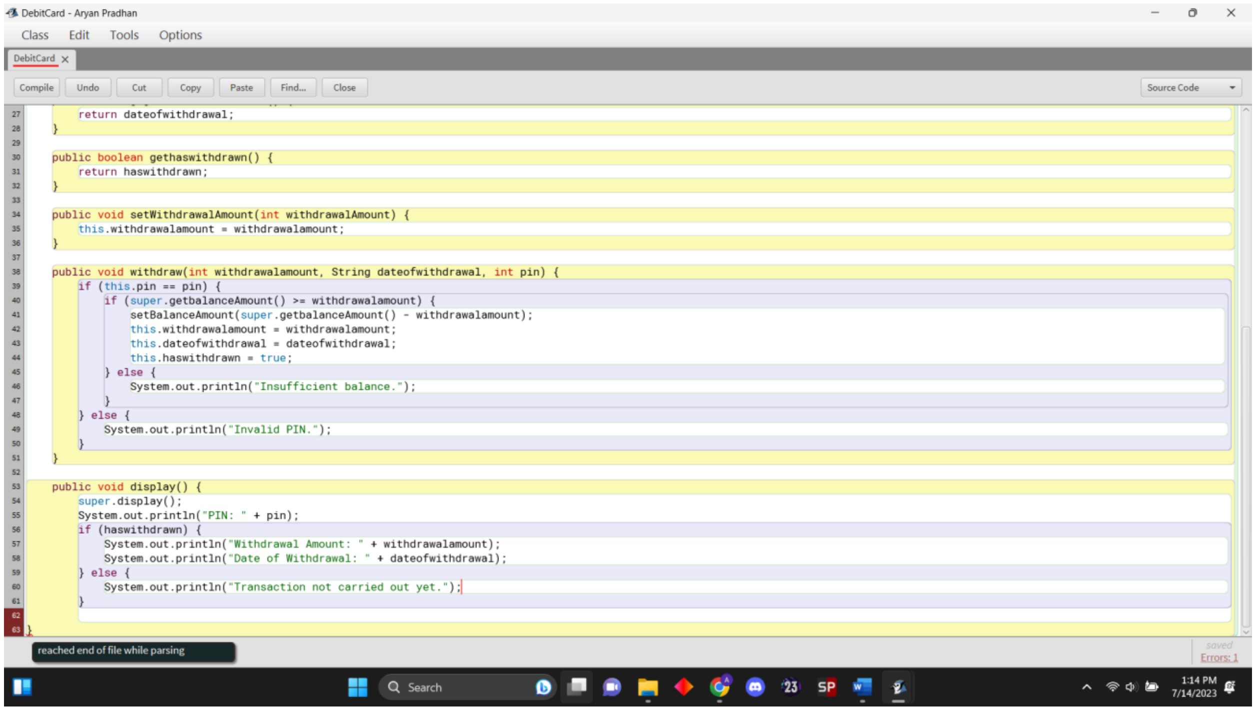
```
Card ID: 1111
Client Name: Aryan Pradhan
Issuer bank: Nabil Bank
Bank account: Saving
Balance amount: 30000.0
PIN: 5555
Transaction not carried out yet.
```

Figure 15

6) Error Detection and Correction

6.1) Syntax Error

In the following figure a small error caused to backspace the closed bracket.

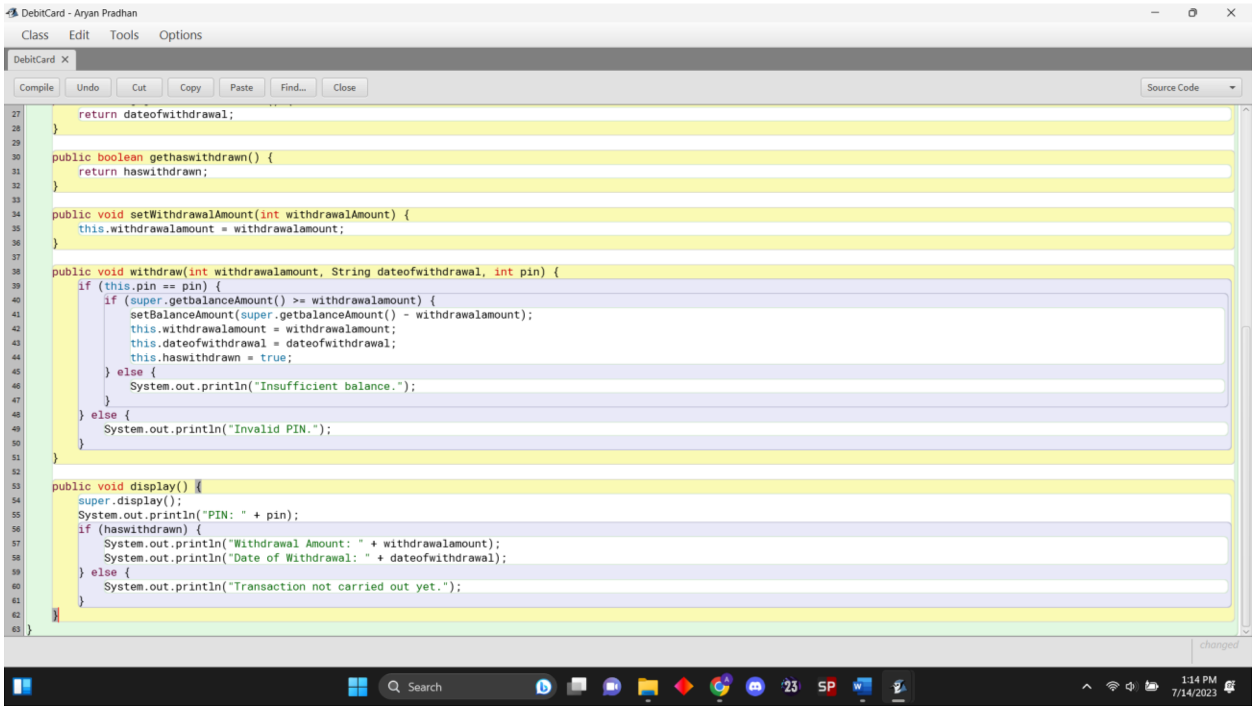


```
        return dateofwithdrawal;
    }

    public boolean gethaswithdrawn() {
        return haswithdrawn;
    }

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

    public void withdraw(int withdrawalamount, String dateofwithdrawal, int pin) {
        if (this.pin == pin) {
            if (super.getbalanceAmount() >= withdrawalamount) {
                setBalanceAmount(super.getbalanceAmount() - withdrawalamount);
                this.withdrawalamount = withdrawalamount;
                this.dateofwithdrawal = dateofwithdrawal;
                this.haswithdrawn = true;
            } else {
                System.out.println("Insufficient balance.");
            }
        } else {
            System.out.println("Invalid PIN.");
        }
    }

    public void display() {
        super.display();
        System.out.println("PIN: " + pin);
        if (haswithdrawn) {
            System.out.println("Withdrawal Amount: " + withdrawalamount);
            System.out.println("Date of Withdrawal: " + dateofwithdrawal);
        } else {
            System.out.println("Transaction not carried out yet.");
        }

}
```

```
reached end of file while parsing
```

Figure 16

But here in the figure the error has been corrected.

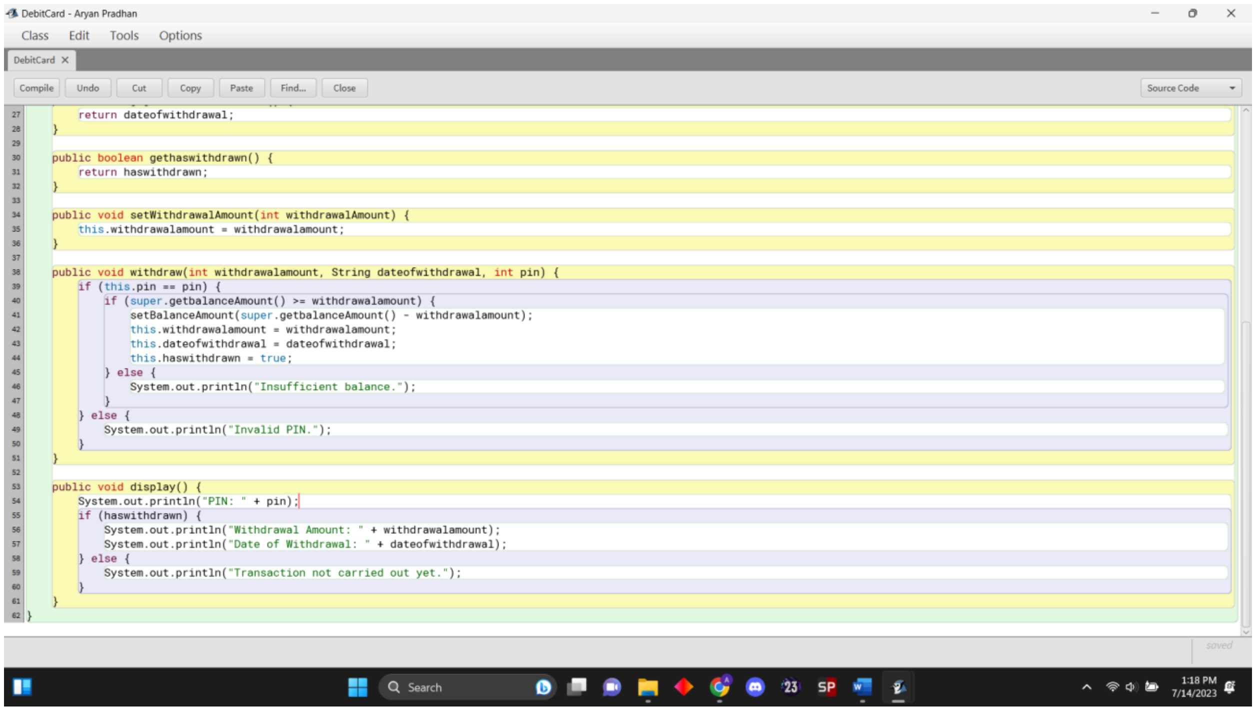


```java
        return dateofwithdrawal;
    }

    public boolean gethaswithdrawn() {
        return haswithdrawn;
    }

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

    public void withdraw(int withdrawalamount, String dateofwithdrawal, int pin) {
        if (this.pin == pin) {
            if (super.getbalanceAmount() >= withdrawalamount) {
                setBalanceAmount(super.getbalanceAmount() - withdrawalamount);
                this.withdrawalamount = withdrawalamount;
                this.dateofwithdrawal = dateofwithdrawal;
                this.haswithdrawn = true;
            } else {
                System.out.println("Insufficient balance.");
            }
        } else {
            System.out.println("Invalid PIN.");
        }
    }

    public void display() {
        super.display();
        System.out.println("PIN: " + pin);
        if (haswithdrawn) {
            System.out.println("Withdrawal Amount: " + withdrawalamount);
            System.out.println("Date of Withdrawal: " + dateofwithdrawal);
        } else {
            System.out.println("Transaction not carried out yet.");
        }
    }
}
```

Figure 17

6.2) Logical Error

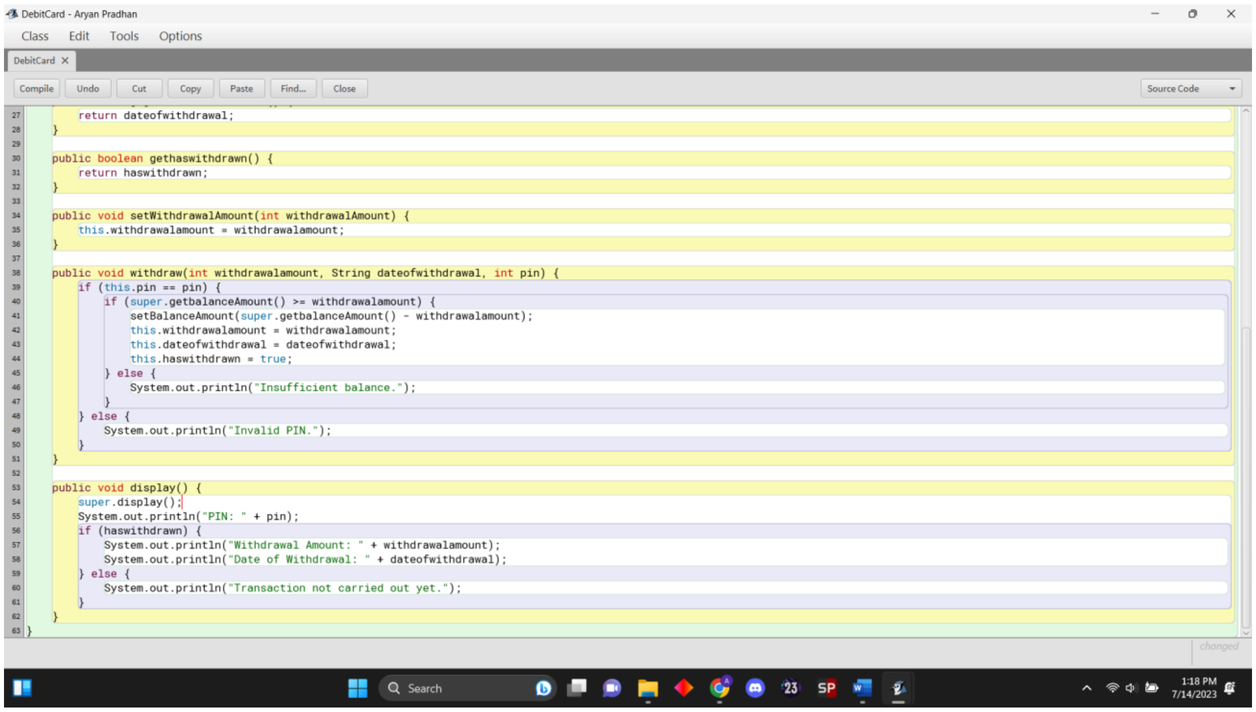
In the following figure an error was caused by the lack of a superclass in the void method.


DebitCard - Aryan Pradhan
Class Edit Tools Options
DebitCard
Compile Undo Cut Copy Paste Find... Close
Source Code
saved

```
        return dateofwithdrawal;
    }

    public boolean gethaswithdrawn() {
        return haswithdrawn;
    }

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

    public void withdraw(int withdrawalamount, String dateofwithdrawal, int pin) {
        if (this.pin == pin) {
            if (super.getbalanceAmount() >= withdrawalamount) {
                setBalanceAmount(super.getbalanceAmount() - withdrawalamount);
                this.withdrawalamount = withdrawalamount;
                this.dateofwithdrawal = dateofwithdrawal;
                this.haswithdrawn = true;
            } else {
                System.out.println("Insufficient balance.");
            }
        } else {
            System.out.println("Invalid PIN.");
        }
    }

    public void display() {
        System.out.println("PIN: " + pin);
        if (haswithdrawn) {
            System.out.println("Withdrawal Amount: " + withdrawalamount);
            System.out.println("Date of Withdrawal: " + dateofwithdrawal);
        } else {
            System.out.println("Transaction not carried out yet.");
        }
    }
}
```

Figure 18

After a while the error was corrected by using super calling the display method from whole another class.

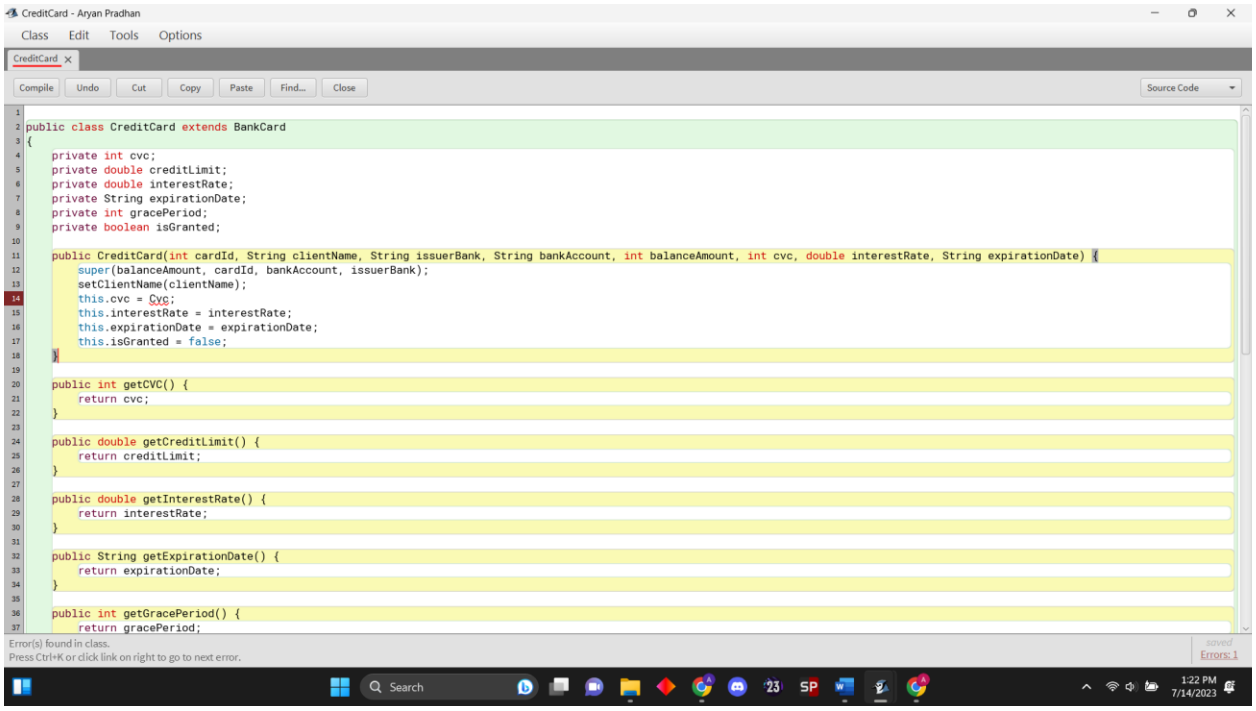


```java
        return dateofwithdrawal;
    }

    public boolean gethaswithdrawn() {
        return haswithdrawn;
    }

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

    public void withdraw(int withdrawalamount, String dateofwithdrawal, int pin) {
        if (this.pin == pin) {
            if (super.getbalanceAmount() >= withdrawalamount) {
                setBalanceAmount(super.getbalanceAmount() - withdrawalamount);
                this.withdrawalamount = withdrawalamount;
                this.dateofwithdrawal = dateofwithdrawal;
                this.haswithdrawn = true;
            } else {
                System.out.println("Insufficient balance.");
            }
        } else {
            System.out.println("Invalid PIN.");
        }
    }

    public void display() {
        super.display();
        System.out.println("PIN: " + pin);
        if (haswithdrawn) {
            System.out.println("Withdrawal Amount: " + withdrawalamount);
            System.out.println("Date of Withdrawal: " + dateofwithdrawal);
        } else {
            System.out.println("Transaction not carried out yet.");
        }
    }
}
```

Figure 19

6.3) Semantic Error

In the following figure an undeclared variable was called.

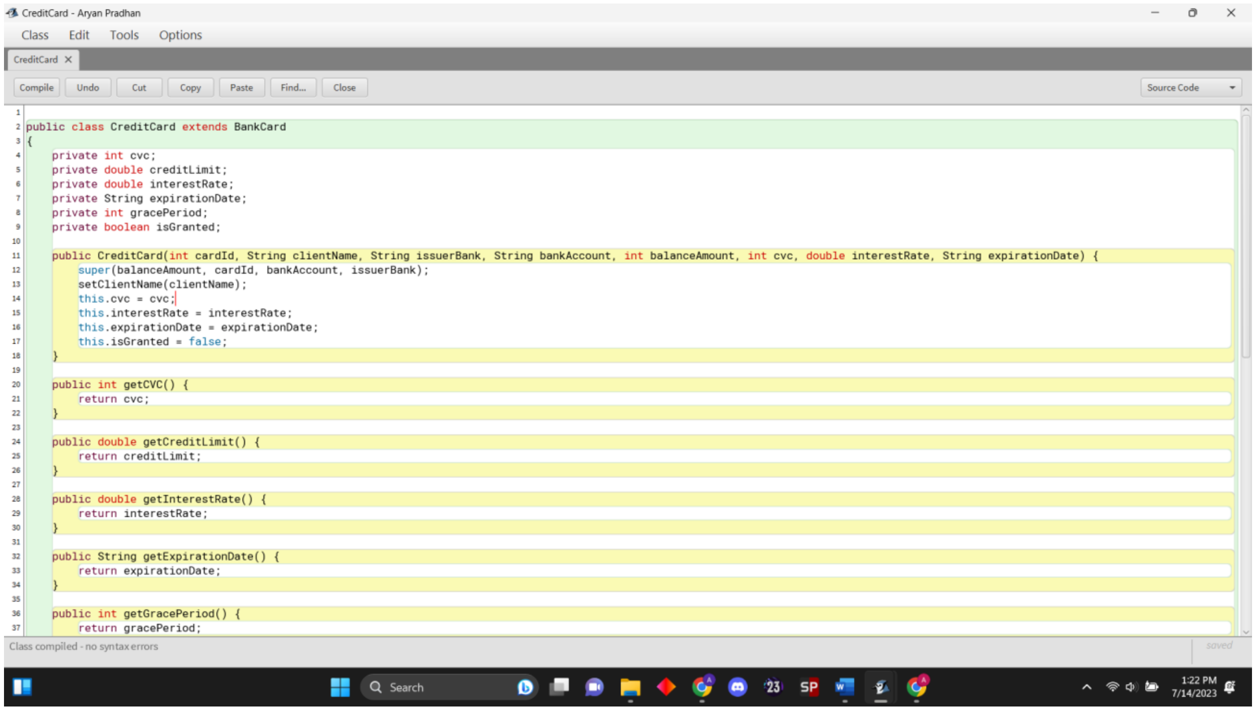

CreditCard - Aryan Pradhan
Class Edit Tools Options
CreditCard
Compile Undo Cut Copy Paste Find... Close
Source Code

```
public class CreditCard extends BankCard
{
    private int cvc;
    private double creditLimit;
    private double interestRate;
    private String expirationDate;
    private int gracePeriod;
    private boolean isGranted;

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

    public int getCVC() {
        return cvc;
    }

    public double getCreditLimit() {
        return creditLimit;
    }

    public double getInterestRate() {
        return interestRate;
    }

    public String getExpirationDate() {
        return expirationDate;
    }

    public int getGracePeriod() {
        return gracePeriod;
```

Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.
saved
Errors: 1

Figure 20

After the correction the proper declared variable was called


CreditCard - Aryan Pradhan
Class Edit Tools Options
CreditCard
Compile Undo Cut Copy Paste Find... Close
Source Code

```java

public class CreditCard extends BankCard
{
    private int cvc;
    private double creditLimit;
    private double interestRate;
    private String expirationDate;
    private int gracePeriod;
    private boolean isGranted;

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

    public int getCVC() {
        return cvc;
    }

    public double getCreditLimit() {
        return creditLimit;
    }

    public double getInterestRate() {
        return interestRate;
    }

    public String getExpirationDate() {
        return expirationDate;
    }

    public int getGracePeriod() {
        return gracePeriod;
```

Class compiled - no syntax errors

Figure 21

7)Conclusion

As per the conclusion, by successfully completing this coursework that demonstrates my ability to apply concepts. This coursework gave me an opportunity to hoe my skills in logical solving and syntax. Completing a project is the beginning of journey as a programmer even though I have a lot of obstacles in my way. These obstacles help me grow and strengthen my skills.

8)Appendix

8.1) Appendix of BankCard

```
public class BankCard

{

    // class attributes

    private int cardId;

    private String clientName;

    private String issuerBank;

    private String bankAccount;

    private double balanceAmount;


    // constructor

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

        this.balanceAmount = balanceAmount;

        this.cardId = cardId;

        this.bankAccount = bankAccount;

        this.issuerBank = issuerBank;

        this.clientName = "";  // initialize clientName to an empty string

    }


    //accessor method

    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;
}

//method to set the client name
public void setClientName(String newClientName)  {
    this.clientName = newClientName;
}

//method to set the balance amount
public void setBalanceAmount(double newBalanceAmount)  {
    this.balanceAmount = newBalanceAmount;
}
```

```
    //method to display card details
    public void display()  {
        if (clientName.isEmpty())  {
            System.out.println("Client name has not been set");
        } else {
            System.out.println("Card ID: "+ cardId);
            System.out.println("Client Name: "+ clientName);
            System.out.println("Issuer bank: "+ issuerBank);
            System.out.println("Bank account: "+ bankAccount);
            System.out.println("Balance amount: "+ balanceAmount);
        }
    }
}
```

8.2) Appendix of DebitCard

```
public class DebitCard extends BankCard
{
    private int pin;
    private int withdrawalamount;
    private String dateofwithdrawal;
    private boolean haswithdrawn;

    public DebitCard(double balanceAmount, int cardId, String bankAccount, String
issuerBank, String clientName, int pin) {
        super(balanceAmount,cardId, bankAccount, issuerBank);
        super.setClientName(clientName);
        super.getclientName();
        this.pin = pin;
        this.haswithdrawn = false;
    }
    //creating accessor method
    public int getpin() {
        return pin;
    }

    public int getwithdrawalamount() {
        return withdrawalamount;
    }

    public String getdateofwithdrawal() {
        return dateofwithdrawal;
    }
```

```
public boolean gethaswithdrawn() {
    return haswithdrawn;
}

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

public void withdraw(int withdrawalamount, String dateofwithdrawal, int pin) {
    if (this.pin == pin) {
        if (super.getbalanceAmount() >= withdrawalamount) {
            setBalanceAmount(super.getbalanceAmount() - withdrawalamount);
            this.withdrawalamount = withdrawalamount;
            this.dateofwithdrawal = dateofwithdrawal;
            this.haswithdrawn = true;
        } else {
            System.out.println("Insufficient balance.");
        }
    } else {
        System.out.println("Invalid PIN.");
    }
}

public void display() {
    super.display();
    System.out.println("PIN: " + pin);
    if (haswithdrawn) {
            System.out.println("Withdrawal Amount: " + withdrawalamount);
            System.out.println("Date of Withdrawal: " + dateofwithdrawal);
        } else {
            System.out.println("Transaction not carried out yet.");
        }
    }
}
```

8.3) Appendix of CreditCard

```
public class CreditCard extends BankCard
{
    private int cvc;
    private double creditLimit;
    private double interestRate;
    private String expirationDate;
    private int gracePeriod;
    private boolean isGranted;

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

    public int getCVC() {
        return cvc;
    }

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

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

public String getExpirationDate() {
    return expirationDate;
}

public int getGracePeriod() {
    return gracePeriod;
}

public boolean getIsGranted() {
    return isGranted;
}

public void setCreditLimit(double creditLimit, int gracePeriod) {
    if (creditLimit <= 2.5 * getbalanceAmount()) {
        this.creditLimit = creditLimit;
        this.gracePeriod = gracePeriod;
        this.isGranted = true;
    } else {
        System.out.println("Credit cannot be issued.");
    }
}

public void cancelCreditCard() {
    if (isGranted) {
            cvc = 0;
            creditLimit = 0;
            gracePeriod = 0;
            isGranted = false;
        } else {
            System.out.println("Invalid Operation");
        }
    }

    public void display() {
        super.display();
        if (isGranted) {
            System.out.println("CVC: " + cvc);
            System.out.println("Credit Limit: " + creditLimit);
            System.out.println("Interest Rate: " + interestRate);
            System.out.println("Expiration Date: " + expirationDate);
            System.out.println("Grace Period: " + gracePeriod);
        } else {
            System.out.println("Credit not granted");
        }
    }
}
```