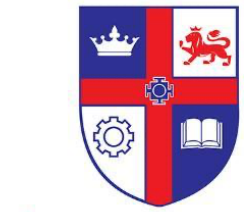
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Pramit Gaha

London Met ID: 22067582

College ID: NP01CP4A220082

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

Table of Figures

Table Of Tables

1. Introduction

Java is a widely-used, class-based, object-oriented programming language. It is well-known for its use in various applications and software development. Object-oriented programming (OOP) is a programming paradigm that uses classes and objects to design programs. The four main concepts of OOP are abstraction, encapsulation, inheritance, and polymorphism.

This coursework provides an overview of a Java program that implements these concepts through the use of classes, such as the BankCard, DebitCard, and CreditCard classes. The BankCard class serves as the parent class and stores information about the client name, card ID, balance, client username, and issuer bank name. The DebitCard class is a child class of BankCard and stores information about the pin number, amount withdrawn, date of withdrawal, and whether or not the card has been withdrawn. The CreditCard class is also a child class of BankCard and stores information about the CVC number, credit limit, date of expiry, grace period, and whether or not the card has been granted. The CreditCard class also includes a method for setting up and canceling a credit card.

The coursework was completed using the BlueJ software, which is a Java development environment. BlueJ is used in various settings, such as software development companies and educational institutions, for creating and testing Java programs. It also allows for tracking the activation status, termination, and expiration of users, as well as detailed information about the users and the medium they are using the software on.

2. Class Diagram

Class diagram of the BankCard and it's child classes DebitCard and Credit Card.

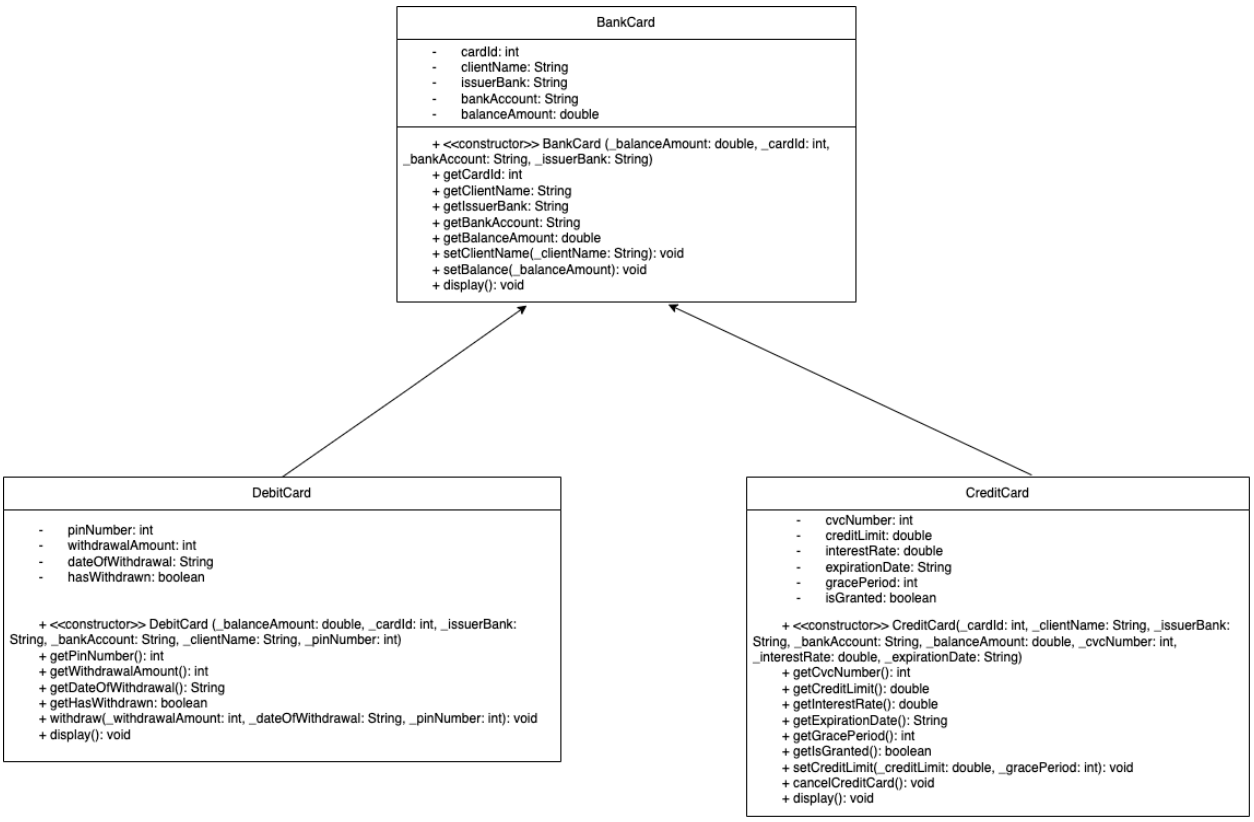


Figure 1 Class Diagram

3. Pseudo Code

3.1. Class BankCard:

3.1.1. CREATE a parent class BankCard

```
DO
    DECLARE instance variable cardId as int
    DECLARE instance variable clientName as string
    DECLARE instance variable issuerBank as string
    DECLARE instance variable bankAccount as string
    DECLARE instance variable balanceAmount as double
END DO
```

3.1.2. CREATE constructor method BankCard with 4 parameters double _balanceAmount, int _cardId, String _bankAccount, String _issuerBank

```
DO
    SET cardId to _cardId
    SET clientName to “”
    SET issuerBank to _issuerBank
    SET bankAccount to _bankAccount
    SET balanceAmount to _balanceAmount
END DO
```

3.1.3. CREATE accessor method getCardId() with return type int

```
DO
    RETURN cardId
END DO
```

3.1.4. CREATE accessor method getClientName() with return type string

```
DO
    RETURN clientName
END DO
```

3.1.5. CREATE accessor method getIssuerBank() with return type string

DO

RETURN issuerBank

END DO

3.1.6. CREATE accessor method getBankAccount() with return type string

DO

RETURN bankAccount

END DO

CREATE accessor method getBalanceAmount() with return type double

DO

RETURN balanceAmount

END DO

CREATE mutator method setClientName with 1 parameter String _clientName

DO

SET clientName to _clientName

END DO

3.1.7. CREATE mutator method setBalance with 1 parameter double _balanceAmount

DO

SET balanceAmount to _balanceAmount

END DO

3.1.8. DEFINE display()

DO

IF clientName is empty

```
            PRINT “Client name isn’t assigned. Please assign a client
            name”
        ELSE
            PRINT "Card Id: " + cardId
            PRINT "Client Name: " + clientName
            PRINT "Issuer Bank: " + issuerBank
            PRINT "Bank Account: " + bankAccount
            PRINT "Balance Amount: " + balanceAmount
        END IF
    END DO
```

3.2. Class DebitCard:

3.2.1. CREATE a child class DebitCard

```
    DO
        DECLARE instance variable pinNumber as int
        DECLARE instance variable withdrawalAmount as int
        DECLARE instance variable dateOfWithdrawal as string
        DECLARE instance variable hasWithdrawn as boolean
    END DO
```

3.2.2. CREATE constructor method DebitCard with 6 parameters double, _balanceAmount, int _cardId, String _issuerBank, String _bankAccount, String _clientName, int _pinNumber

```
    DO
        CALL made to superclass constructor method with 4 values
        _balanceAmount, _cardId, _bankAccount, _issuerBank
        CALL made to superclass setClientName method with 1 value
        _clientName
        SET pinNumber to _pinNumber
        SET hasWithdrawn to false
    END DO
```

3.2.3. CREATE accessor method getPinNumber() with return type int
DO
RETURN pinNumber
END DO

3.2.4. CREATE accessor method getWithdrawalAmount() with return type int
DO
RETURN withdrawalAmount
END DO

3.2.5. CREATE accessor method getDateOfWithdrawal() with return type string
DO
RETURN dateOfWithdrawal
END DO

3.2.6. CREATE accessor method getHasWithdrawn with return type boolean
DO
RETURN hasWithdrawn
END DO

3.2.7. CREATE mutator method Withdraw with 3 parameters int _withdrawalAmount, String _dateOfWithdrawal, int _pinNumber
DO
IF pinNumber is not equal to _pinNumber
Print “Wrong pin. Enter your pin again”
ELSE
DECLARE a variable named currentBalance as double
SET currentBalance to the value returned from the call made to superclass getBalanceAmount method
IF _withdrawalAmount is greater than currentBalance
Print “Insufficient Balance”

ELSE

DECLARE a variable named newBalance as double

SET newBalance to the value got from subtraction of currentBalance and _withdrawalAmount

CALL made to superclass setBalance method with value newBalance

SET hasWithdrawn to true

SET withdrawalAmount to _withdrawalAmount

SET dateOfWithdrawal to _dateOfWithdrawal

END IF

END IF

END DO

3.2.8. DEFINE display()

DO

CALL made to superclass display method

Print “Pin number: “ + pinNumber

IF hasWithdrawn equals to true

Print "Withdrawal Amount: " + withdrawalAmount

Print "Date of Withdrawal: " + dateOfWithdrawal

END IF

END DO

3.3. Class CreditCard:

3.3.1. CREATE a child class CreditCard

DO

DECLARE instance variable cvcNumber as int

DECLARE instance variable creditLimit as double

DECLARE instance variable interestRate as double

DECLARE instance variable expirationDate as string

DECLARE instance variable gracePeriod as int

DECLARE instance variable isGranted as boolean

END DO

3.3.2. CREATE constructor method CreditCard with 8 parameter int _cardId, String _clientName, String _issuerBank, String _bankAccount, double _balanceAmount, int _cvcNumber, double _interestRate, String _expirationDate

DO

CALL made to superclass constructor method with 4 values _balanceAmount, _cardId, _bankAccount, _issuerBank

CALL made to superclass setClientName method with 1 value _clientName

SET cvcNumber to _cvcNumber

SET interestRate to _interestRate

SET expirationDate to _expirationDate

SET isGranted to false

END DO

3.3.3. CREATE an accessor method getCvcNumber() with return type of int

DO

RETURN cvcNumber

END DO

3.3.4. CREATE an accessor method getCreditLimit() with return type of double

DO

RETURN creditLimit

END DO

3.3.5. CREATE an accessor method getInterestRate() with return type of double

DO

RETURN interestRate

END DO

3.3.6. CREATE an accessor method getExpirationDate() with return type of string

DO

RETURN expirationDate

END DO

3.3.7. CREATE an accessor method getGracePeriod() with return type of int

DO

RETURN gracePeriod

END DO

3.3.8. CREATE an accessor method getIsGranted() with return type of boolean

DO

RETURN isGranted

END DO

3.3.9. CREATE a mutator method setCreditLimit with 2 parameter double _creditLimit, int _gracePeriod

DO

DECLARE a variable named allowedCreditLimit as double

SET allowedCreditLimit to the value got from multiplication of 2.5 to the value got from CALL made to superclass getBalanceAmount method

IF _creditLimit is less than or equal to allowedCreditLimit

SET creditLimit to _creditLimit

SET gracePeriod to _gracePeriod

SET isGranted to true

ELSE

Print “Credit limit is higher than the allowed limit, Please input a lower credit limit than " + allowedCreditLimit

END IF

END DO

3.3.10. CREATE a mutator method cancelCreditCard

DO

SET cvcNumber to zero

SET creditLimit to zero

SET gracePeriod to zero

SET isGranted to false

END DO

3.3.11. DEFINE display

DO

IF isGranted is equal to true

CALL made to superclass display method

Print “CVC Number: “ + cvcNumber

Print “Credit Limit: “ + creditLimit

Print “Interest Rate: “ + interestRate

Print “Expiration Date: “ + expirationDate

Print “Grace Period: “ + gracePeriod

ELSE

Print “Credit has not been granted. Credit Limit and Grace Period not available.”

END IF

END DO

4. Method Description

The method of parent class BankCard and the child classes: DebitCard and CreditCard have been described clearly below.

4.1. Class BankCard: (JavaTpoint, 2021) (Mendeley, 2022) (GeeksForGeeks, 16 Jan, 2023) (JavaTpoint, 2021) (GeeksForGeeks, 16 Jan, 2023)

4.1.1. public int getCardId:

This function returns the ID of the card.

4.1.2. public String getClientName:

This function returns the name of the person who holds the card.

4.1.3. public String getIssuerBank:

This function returns the name of the bank that issued the card.

4.1.4. public String getBankAccount:

This function returns the bank account associated with the card.

4.1.5. public double getBalanceAmount:

This function returns the current balance of the card.

4.1.6. public void setClientName:

This function sets the name of the person who holds the card.

4.1.7. public void setBalance:

This function sets the new balance of the card.

4.1.8. public void display:

This function prints the card details, such as ID, holder's name, issuer bank, bank account, and balance. However, there is one condition that if the client’s name is empty a message about setting up the client’s name is displayed.

4.2. Class DebitCard:

4.2.1. public int getPinNumber:

This function returns the pin number of the card.

4.2.2. public int getWithdrawalAmount:

This function returns the withdrawal amount of the card.

4.2.3. public String getDateOfWithdrawal:

This function returns the date of withdrawal of the card.

4.2.4. public boolean getHasWithdrawn:

This function returns a boolean value to check if the card has been withdrawn or not. If the result is true, it means withdrawal activities have happened.

4.2.5. public void withdraw:

This function is used to withdraw money from the card. It checks for the correct pin number and whether the card has enough balance or not. If the conditions are met, it updates the balance amount and the withdrawal amount, date and sets hasWithdrawn to true.

4.2.6. public void display:

This function displays the details of the debit card, including the pin number, withdrawal amount, date of withdrawal, and other information inherited from the BankCard class.

4.3. Class CreditCard:

4.3.1. public int getCvcNumber:

This function returns the CVC number of the credit card.

4.3.2. public double getCreditLimit:

This function returns the credit limit of the credit card.

4.3.3. public double getInterestRate:

This function returns the interest rate of the credit card.

4.3.4. public String getExpirationDate:

This function returns the expiration date of the credit card.

4.3.5. public int getGracePeriod:

This function returns the grace period of the credit card.

4.3.6. public boolean getIsGranted:

This function returns a boolean indicating whether credit has been granted for the credit card.

4.3.7. public void setCreditLimit:

This function sets the credit limit and grace period for the credit card if the credit limit is less than or equal to 2.5 times the balance amount in the account.

4.3.8. public void cancelCreditCard:

This function cancels the credit card by setting all the credit-related fields to zero and setting isGranted to false.

4.3.9. public void display:

This function displays the details of the credit card, including the CVC number, credit limit, interest rate, expiration date, grace period and other information inherited from the BankCard class.

5. Testing

5.1. Testing 1 – To inspect the DebitCard class, withdraw the amount, and re-inspect the DebitCard Class

Table 1. To inspect the DebitCard class, withdraw the amount, and re-inspect the DebitCard Class

Test no:	**1**
Objective:	To inspect the DebitCard class, withdraw the amount, and re-inspect the DebitCard Class.
Action:	• The DebitCard is called with following arguments: _balanceAmount=4000 _cardId=23456 _issuerBank=”NIC Asia” _bankAccount=”pramit21” _clientName=”Pramit Gaha” _pinNumber=2001 • Inspection of DebitCard class. • void withdraw is called with following arguments: _withdrawalAmount=1000 _dateOfWithdrawal=”25-01-2023” _pinNumber=2001 • Re-inspection of DebitCard class.
Expected Result:	Amount will be deducted from balance.
Actual Result:	Amount was deducted from balance.
Conclusion:	The test was successful.

Output Result:

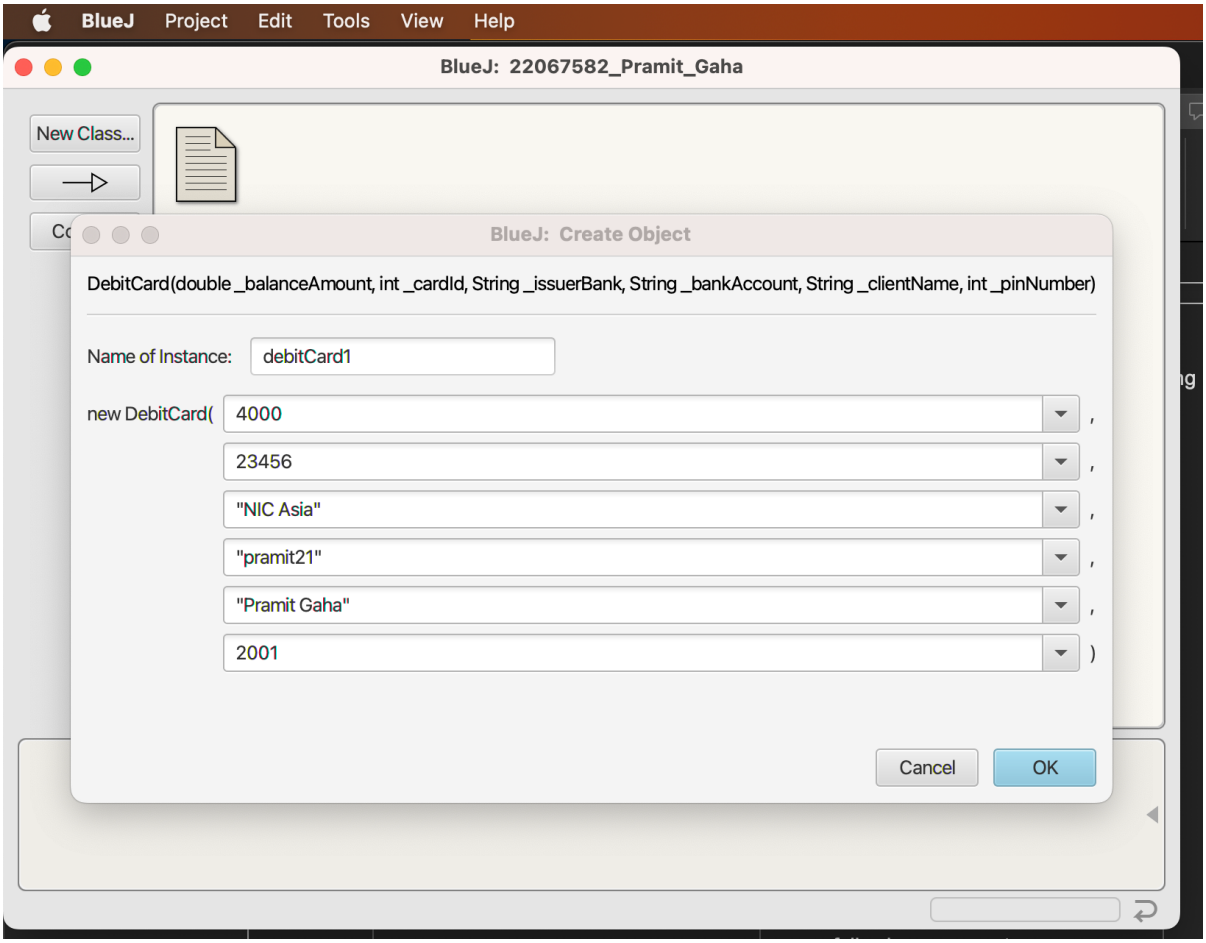


Figure 2. Screenshot of assigning the data in DebitCard class

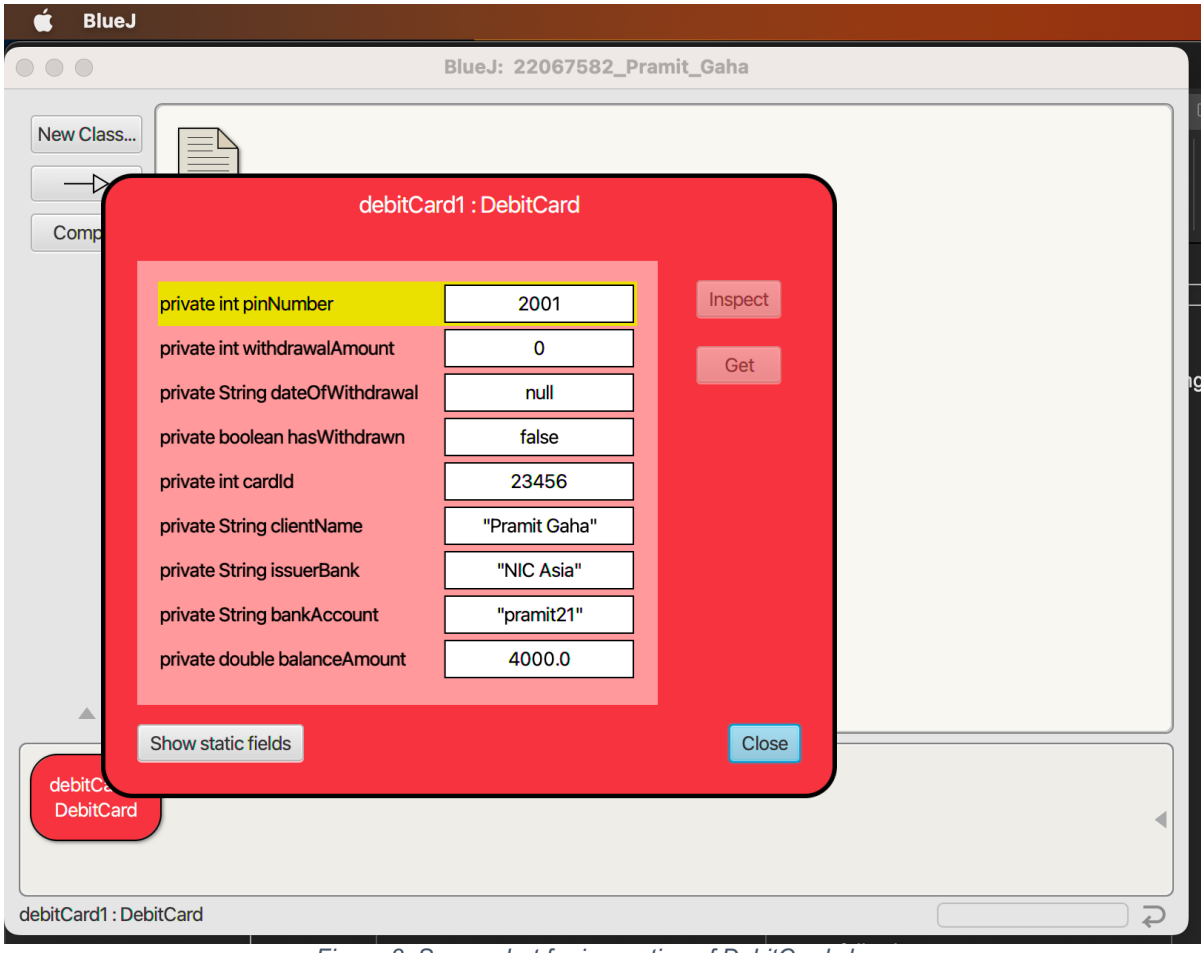


Figure 3. Screenshot for inspection of DebitCard class

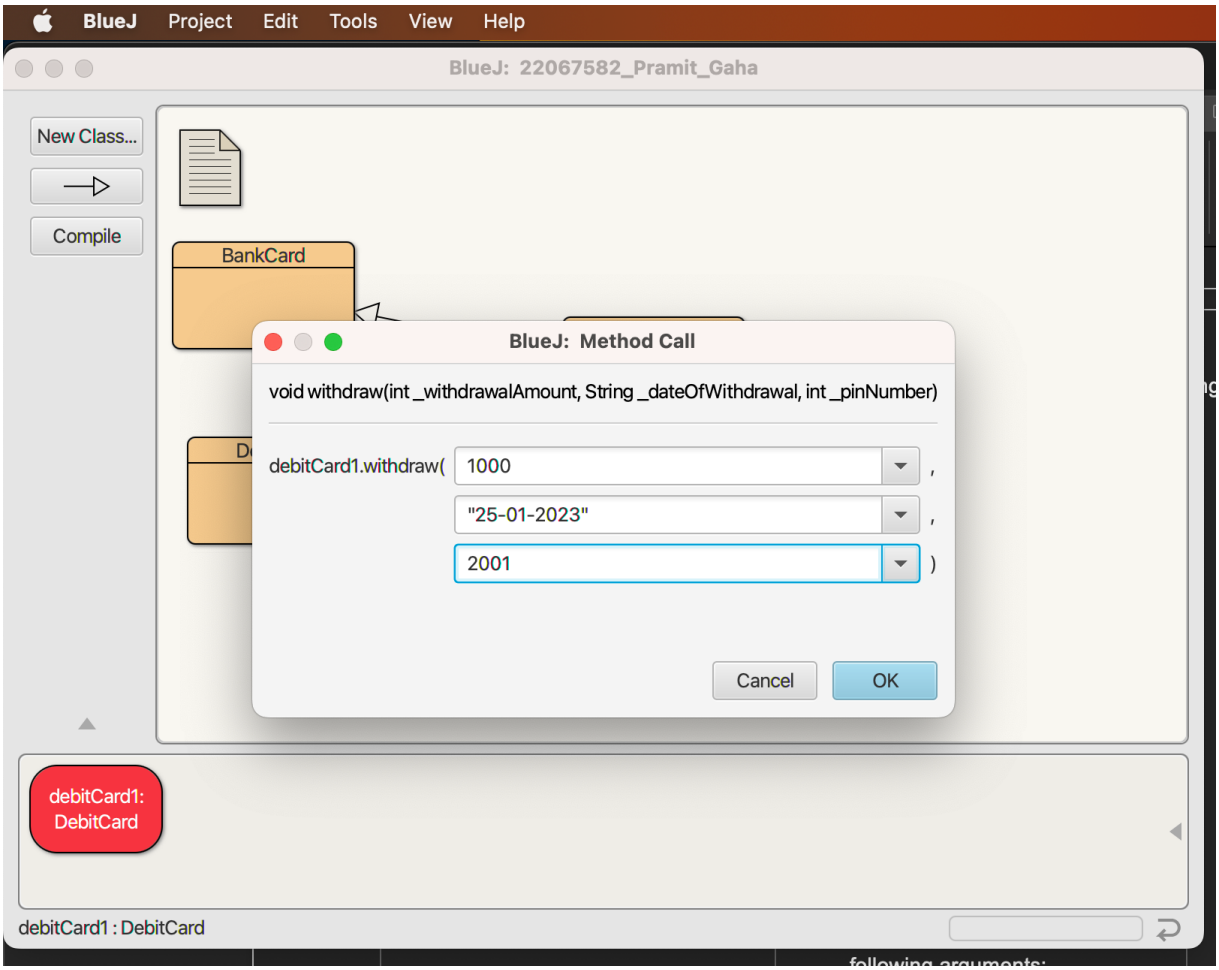


Figure 4. Screenshot for assigning data to withdraw function

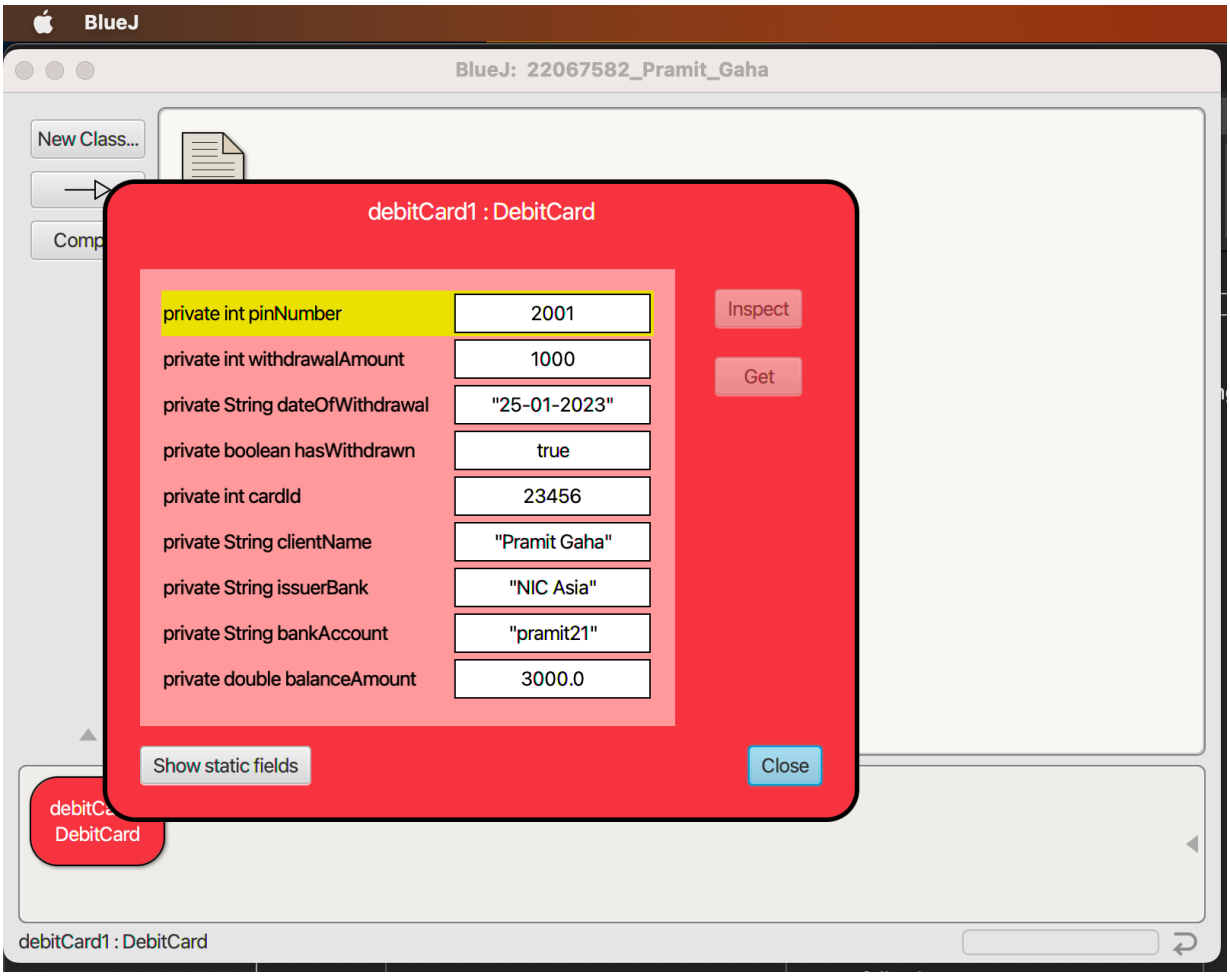


Figure 5. Screenshot for inspection of DebitCard Class

5.2. Test 2 – To inspect CreditCard class, set the credit limit and re-inspect the CreditCard class

Table 2. To inspect the CreditCard class, set the credit limit and re-inspect the CreditCard Class

Test No.	2
Objective:	To inspect CreditCard class, set the credit limit and re-inspect the CreditCard Class
Action:	• The CreditCard is called with the following arguments: _cardId=23456 _clientName=”Pramit Gaha” _issuerBank=”NIC Asia” bankAccount=”pramit21” _balanceAmount=3000 _cvcNumber=232 _interestRate=11 _expirationDate=”01-01-2028” • Inspection of CreditCard Class • void setCreditLimit is called with the following arguments: _creditLimit=6000 _gracePeriod=60 • Re-inspection of CreditCard Class.
Expected Result:	Credit limit would be set.
Actual Result:	Credit limit was set.
Conclusion:	The test was successful

Output Result:

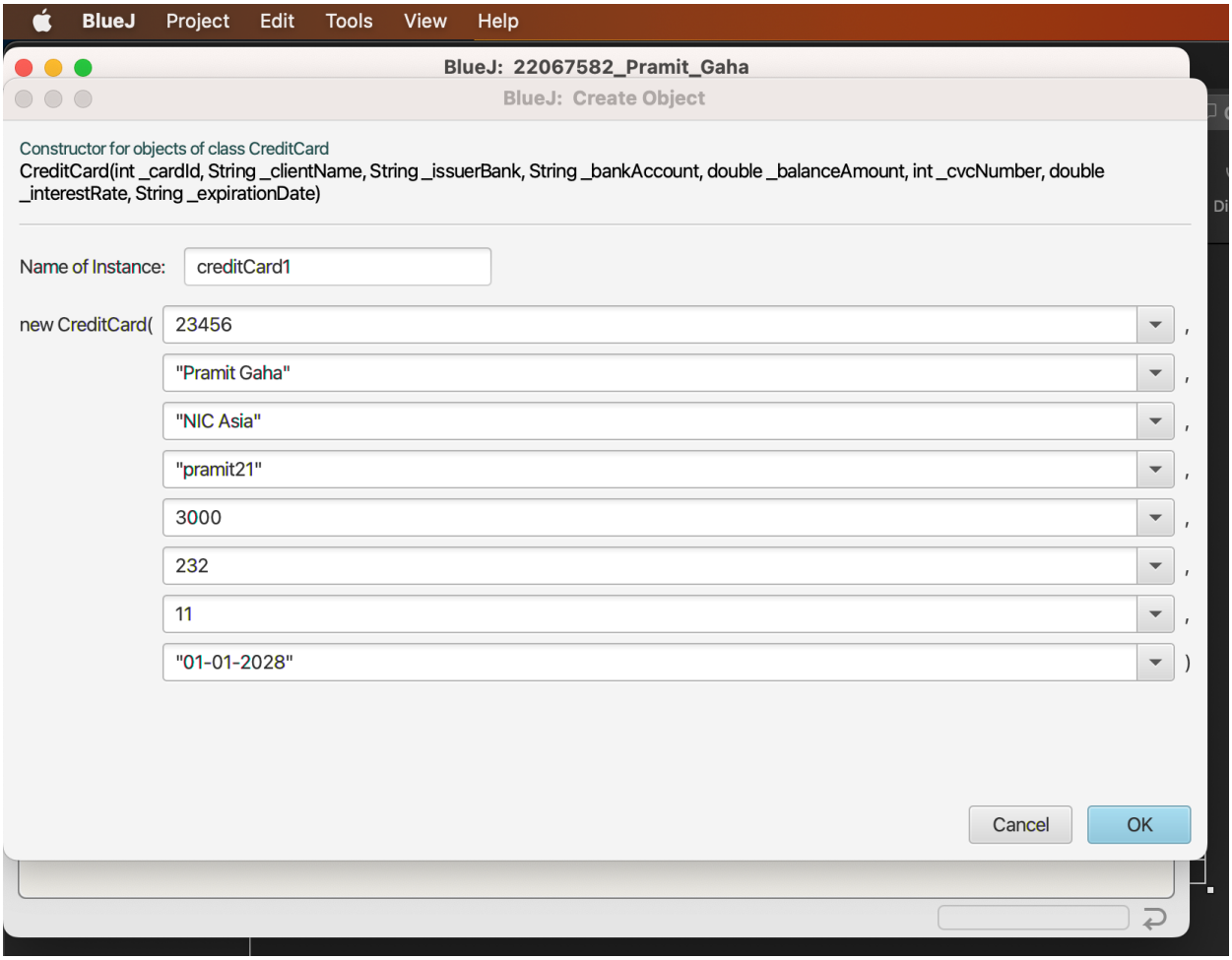


Figure 6. Screenshot for assigning data in the CreditCard Class

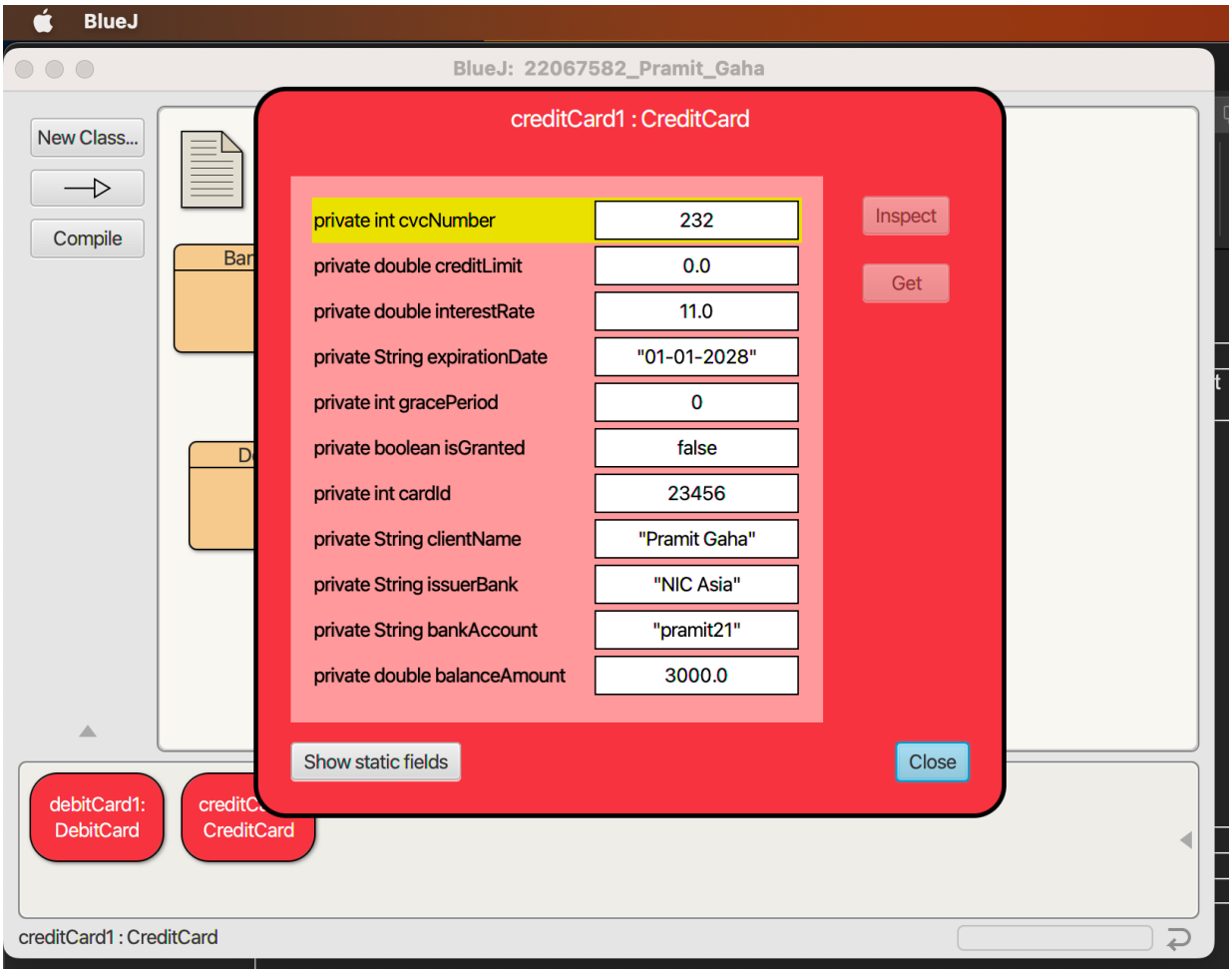


Figure 7. Screenshot for inspection of CreditCard Class

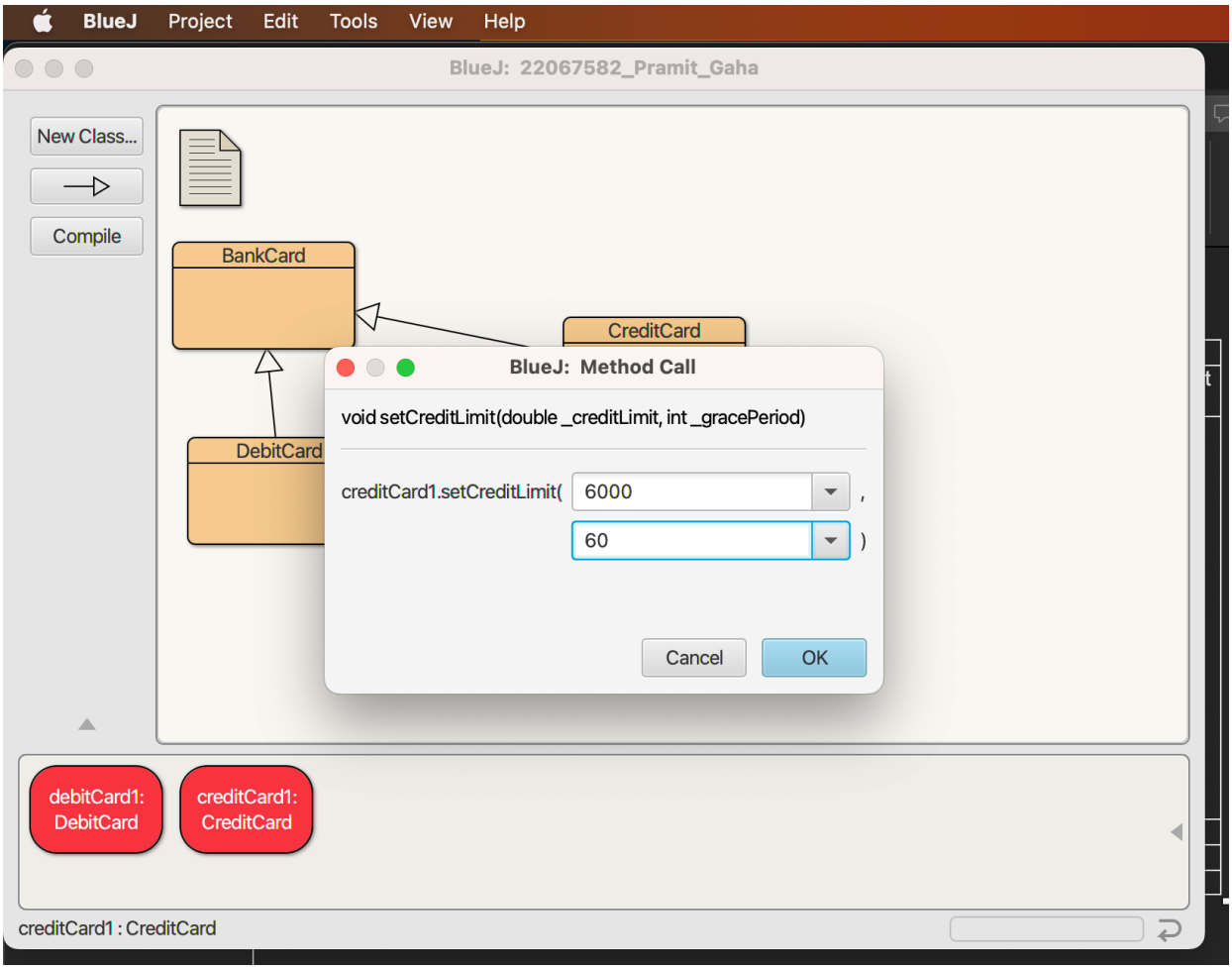


Figure 8. Screenshot for calling setCreditLimit method

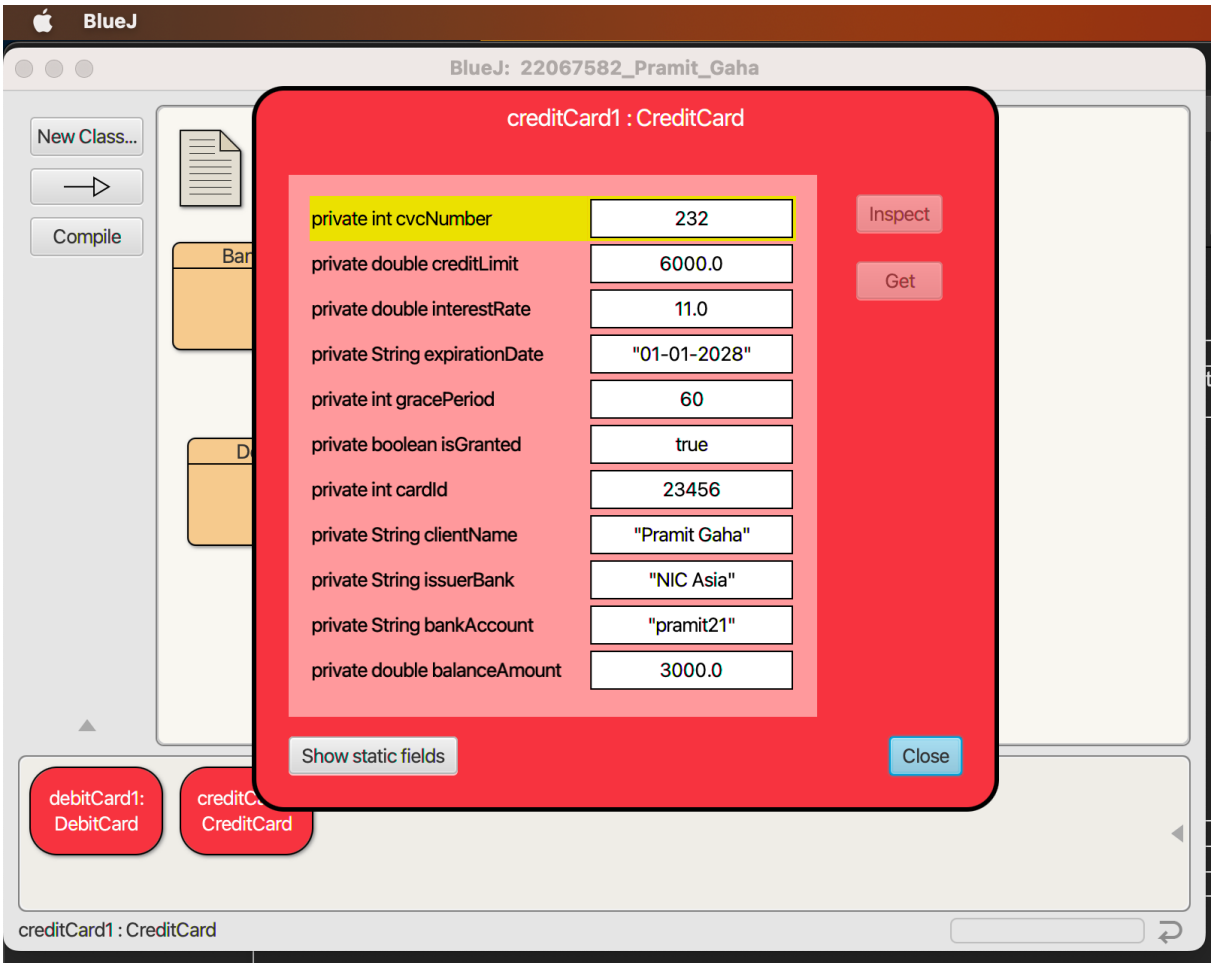


Figure 9. Screenshot for re-inspection of CreditCard Class

5.3. Test 3 – To inspect CreditCard Class again after cancelling the credit card

Table 3. To inspect the CreditCard Class again after cancelling the credit card

Test No.	3
Objective:	To inspect CreditCard Class again after cancelling the credit card.
Actions:	• void cancelCreditCard was called. • Inspection of CreditCard Class.
Expected Result:	Credit card would be cancelled.
Actual Result:	Credit card was cancelled.
Conclusion:	The result of test was successful.

Output Result:

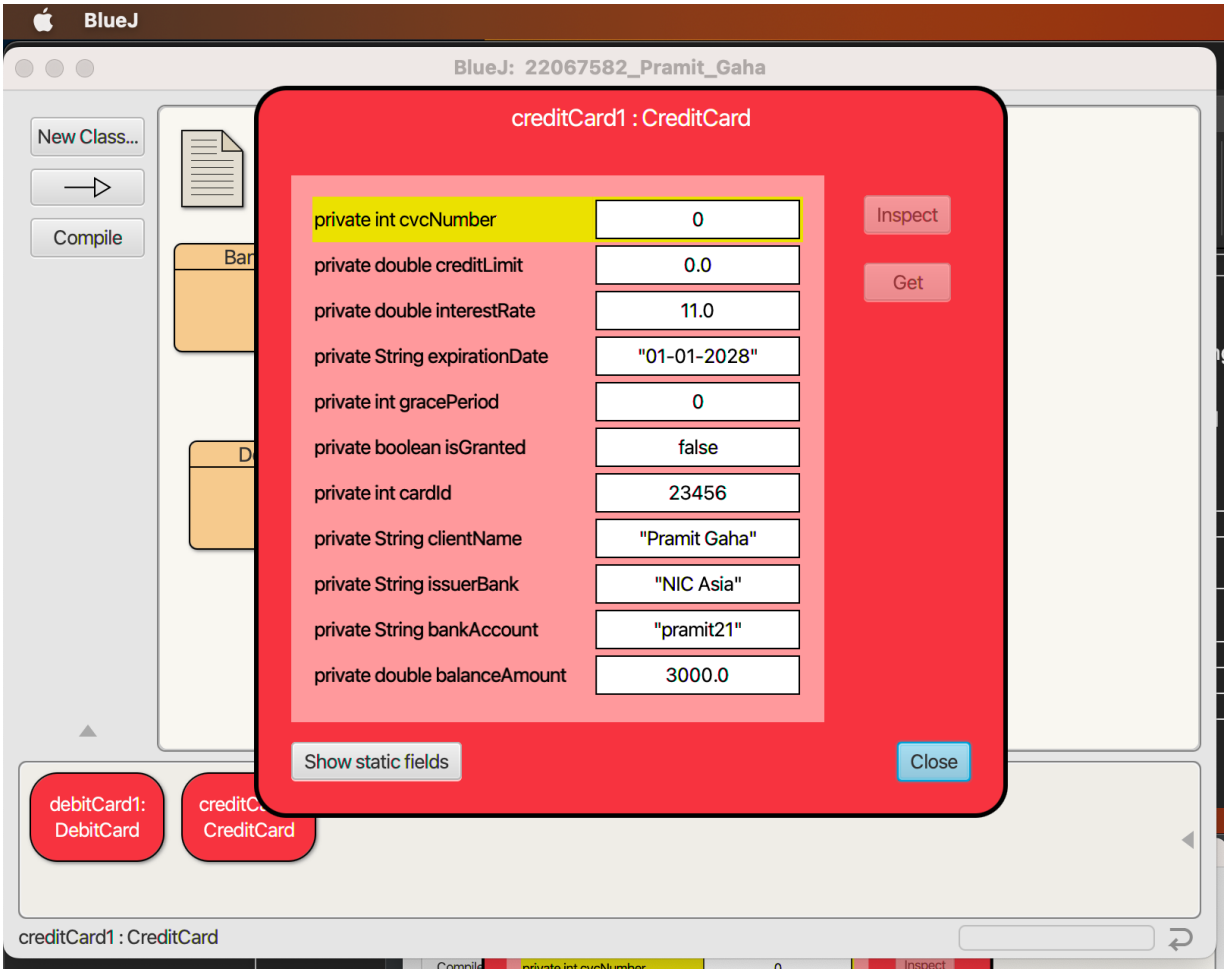


Figure 10. Screenshot for inspection of CreditCard Class

5.4. Test 4 – To display the details of DebitCard and CreditCard classes

Table 4. To display the details of DebitCard and CreditCard class

Test No.	4
Objective:	To display the details of DebitCard and CreditCard classes
Actions:	• void display of DebitCard was called. • void display of CreditCard was called.
Expected Result:	Details of DebitCard and CreditCard classes would be displayed.
Actual Result:	Details of DebitCard and CreditCard classes was displayed.
Conclusion:	The result of test was successful.

Output Result:

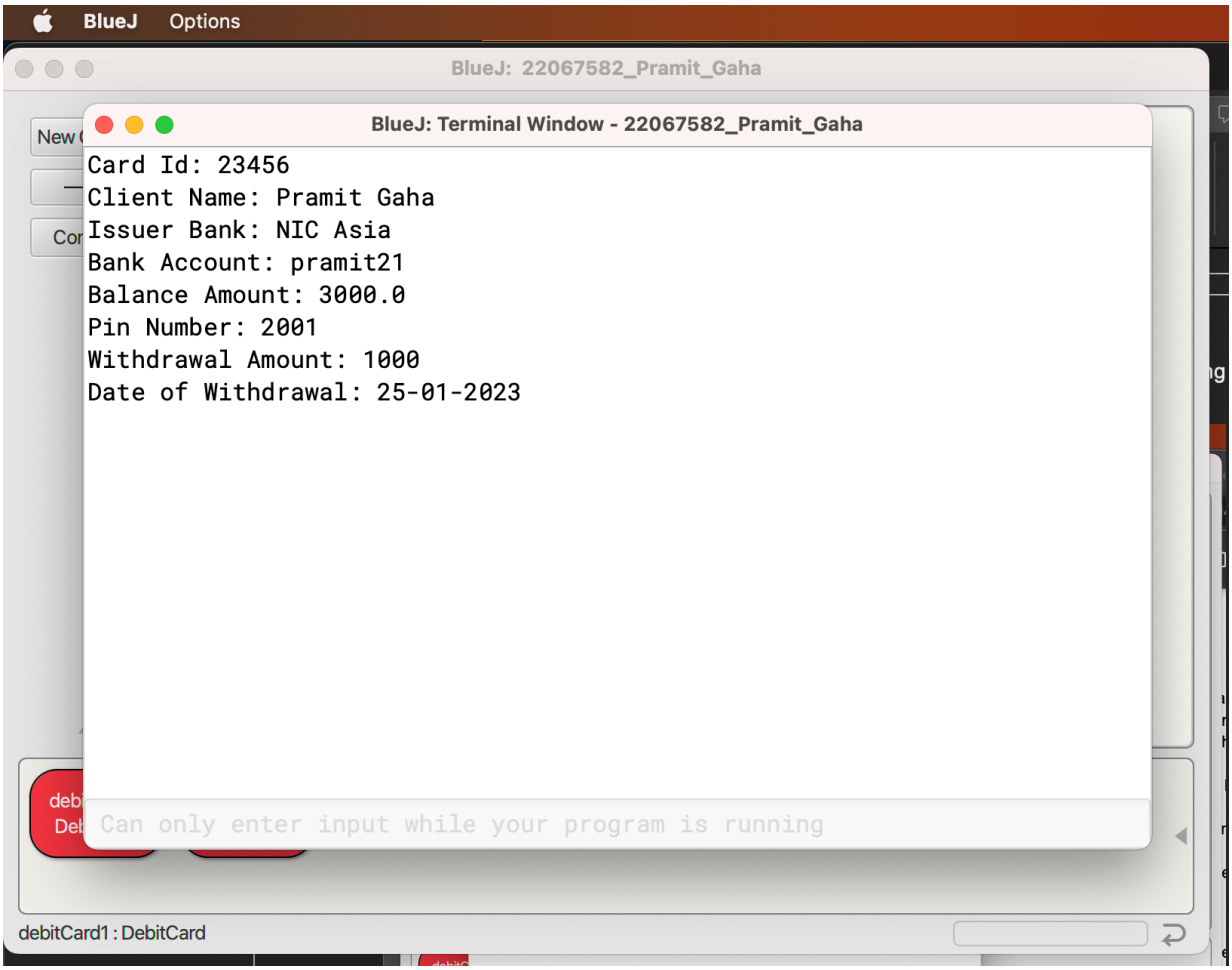


Figure 11. Screenshot for details of DebitCard

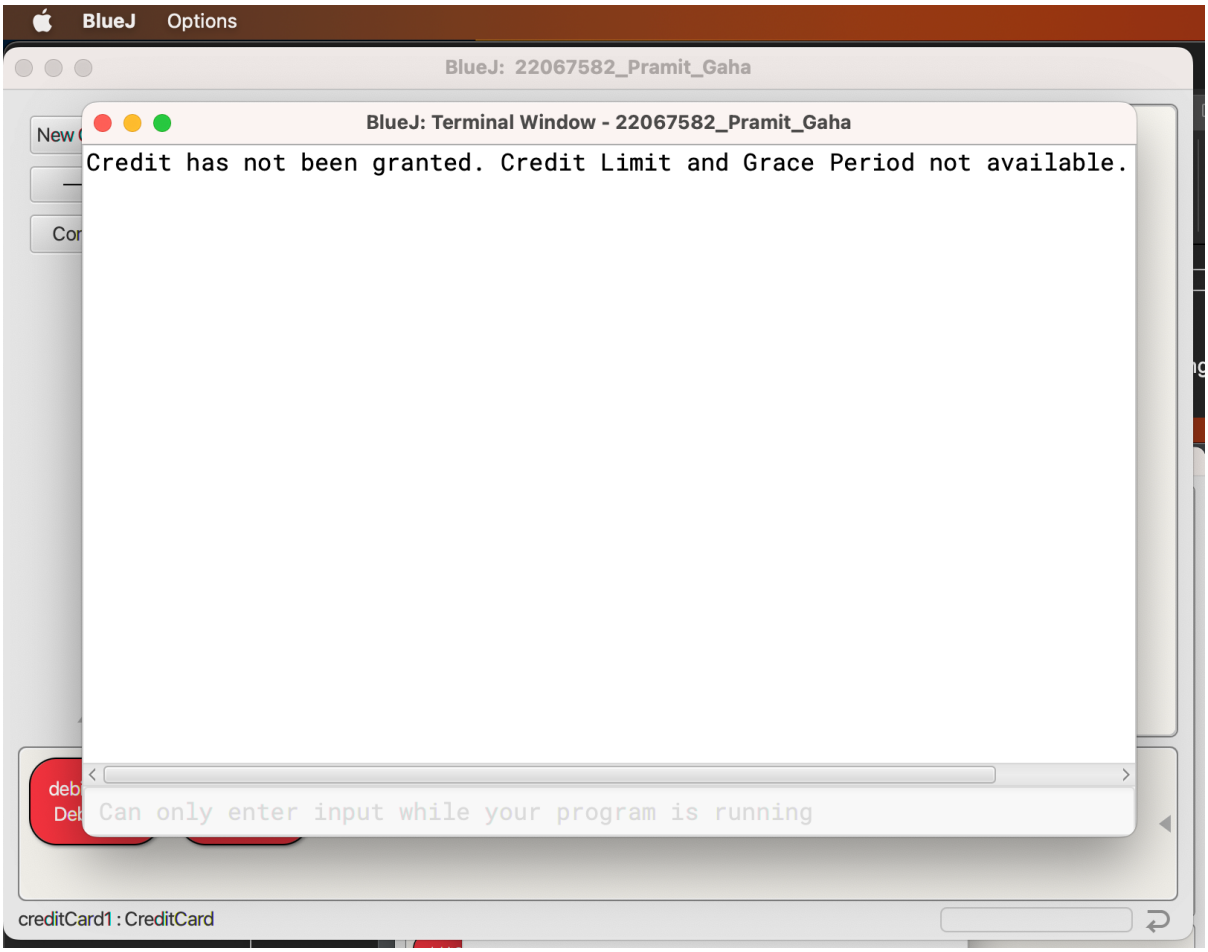


Figure 12. Screenshot for the output of calling display method of CreditCard

6. Error Detection and Correction

6.1. Error 1

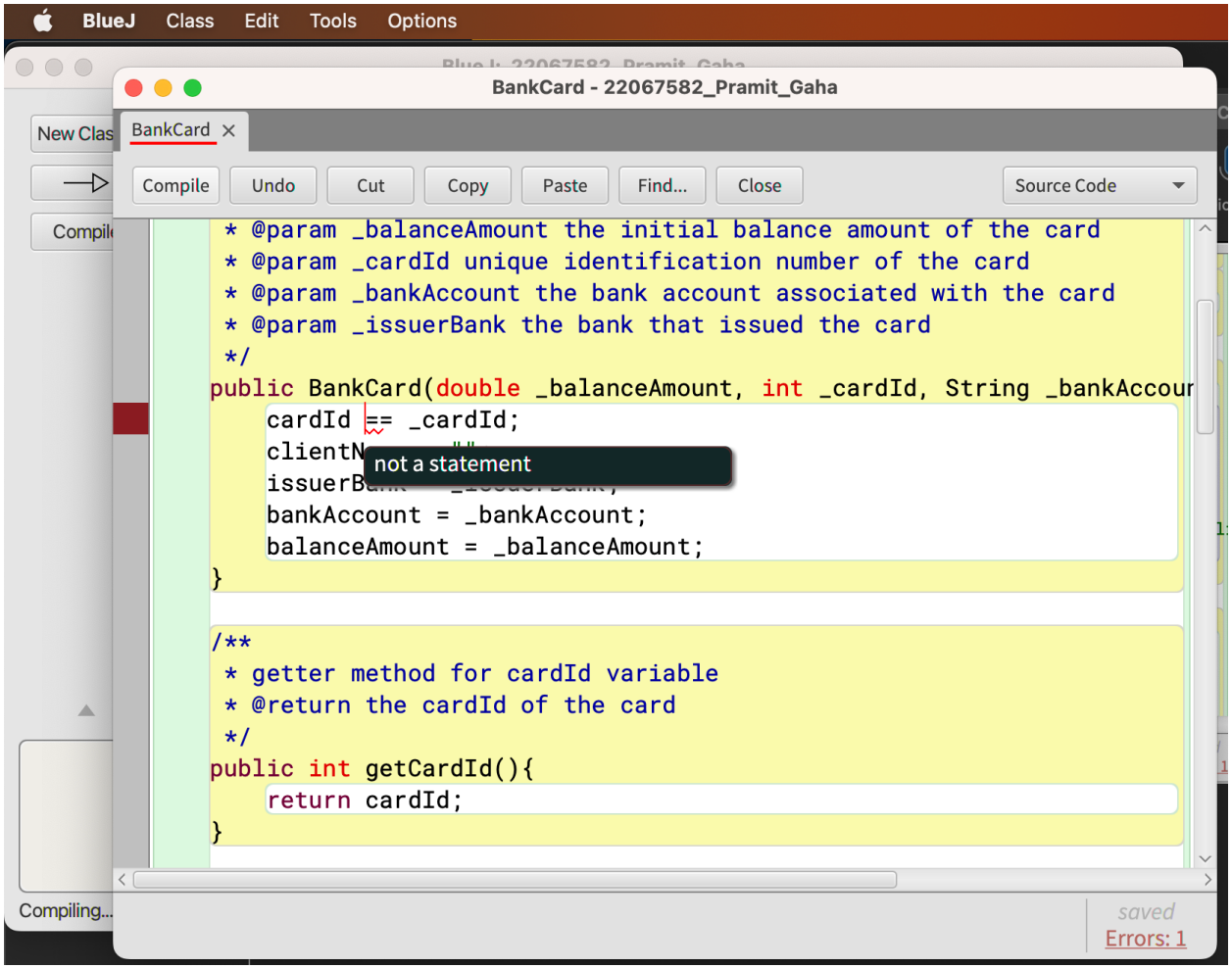


Figure 13. Screenshot for error 1

In the above shown screenshot, there was an error in line cardId == _cardId. To solve the error, cardId == _cardId was replaced by cardId = _cardId. The error was caused due to an extra = in the line. The “=” is used for assigning value whereas “==” is used for comparison.

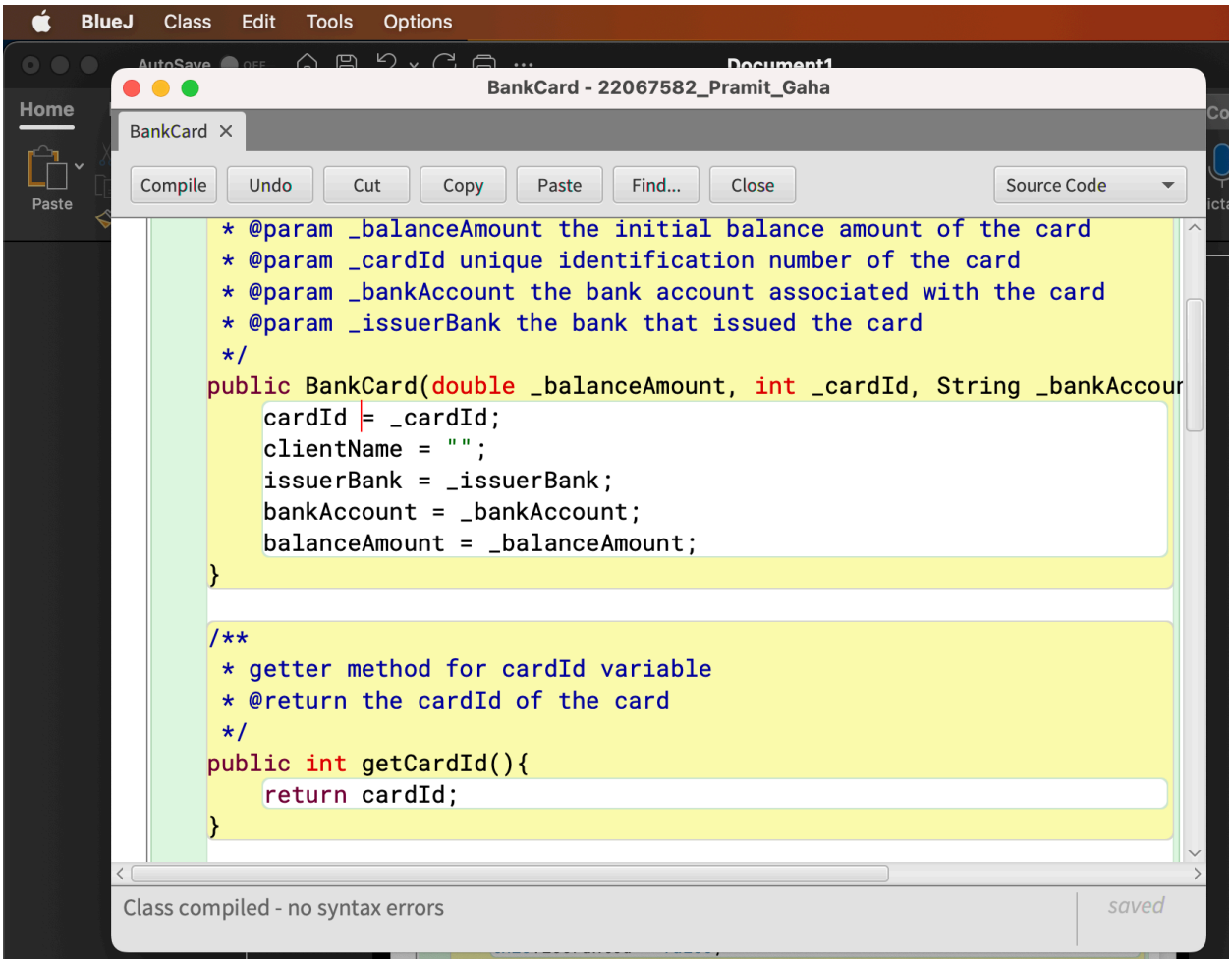


Figure 14. Screenshot after fixing the error 1

6.2. Error 2

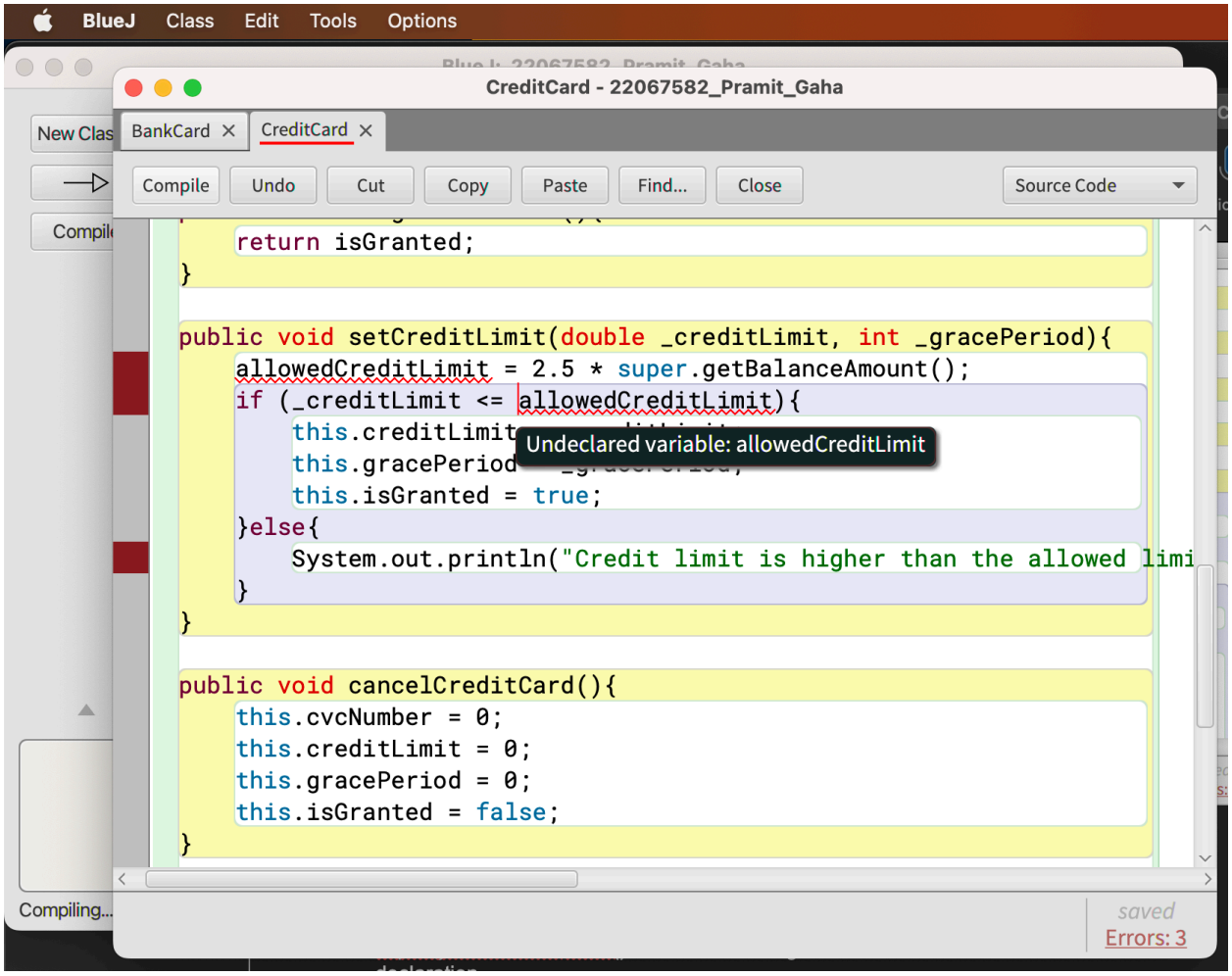


Figure 15. Screenshot for Error 2

In the above shown screeshot, there were errors in 3 lines and the error was Undeclared variable: AllowedCreditLimit. This error was fixed by replacing allowedCreditLimit = 2.5 * super.getBalanceAmount(); line with double allowedCreditLimit = 2.5 * super.getBalanceAmount(); . The error was generated due use of variable without declaration.

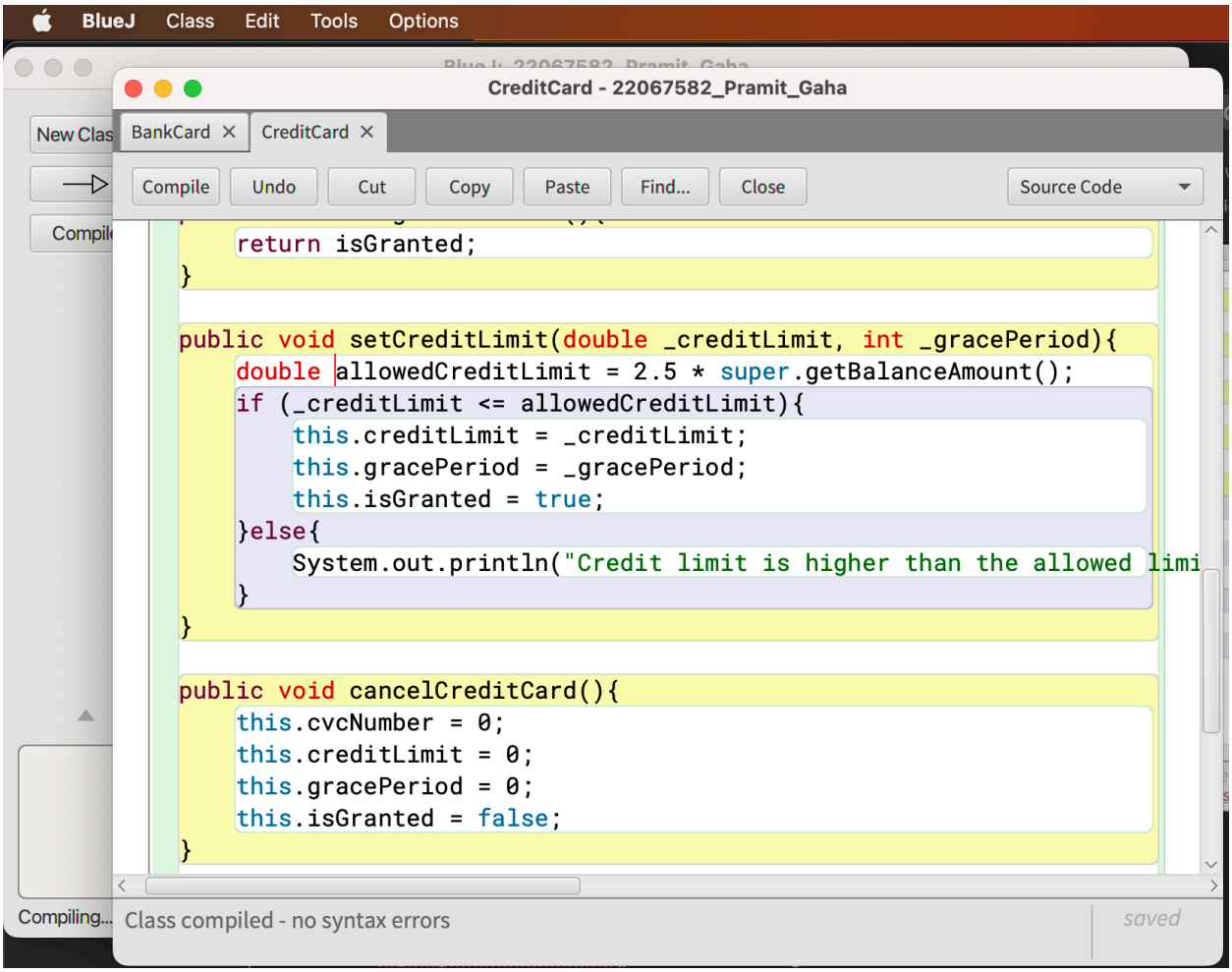


Figure 16. Screenshot for fixing the error 2

6.3. Error 3

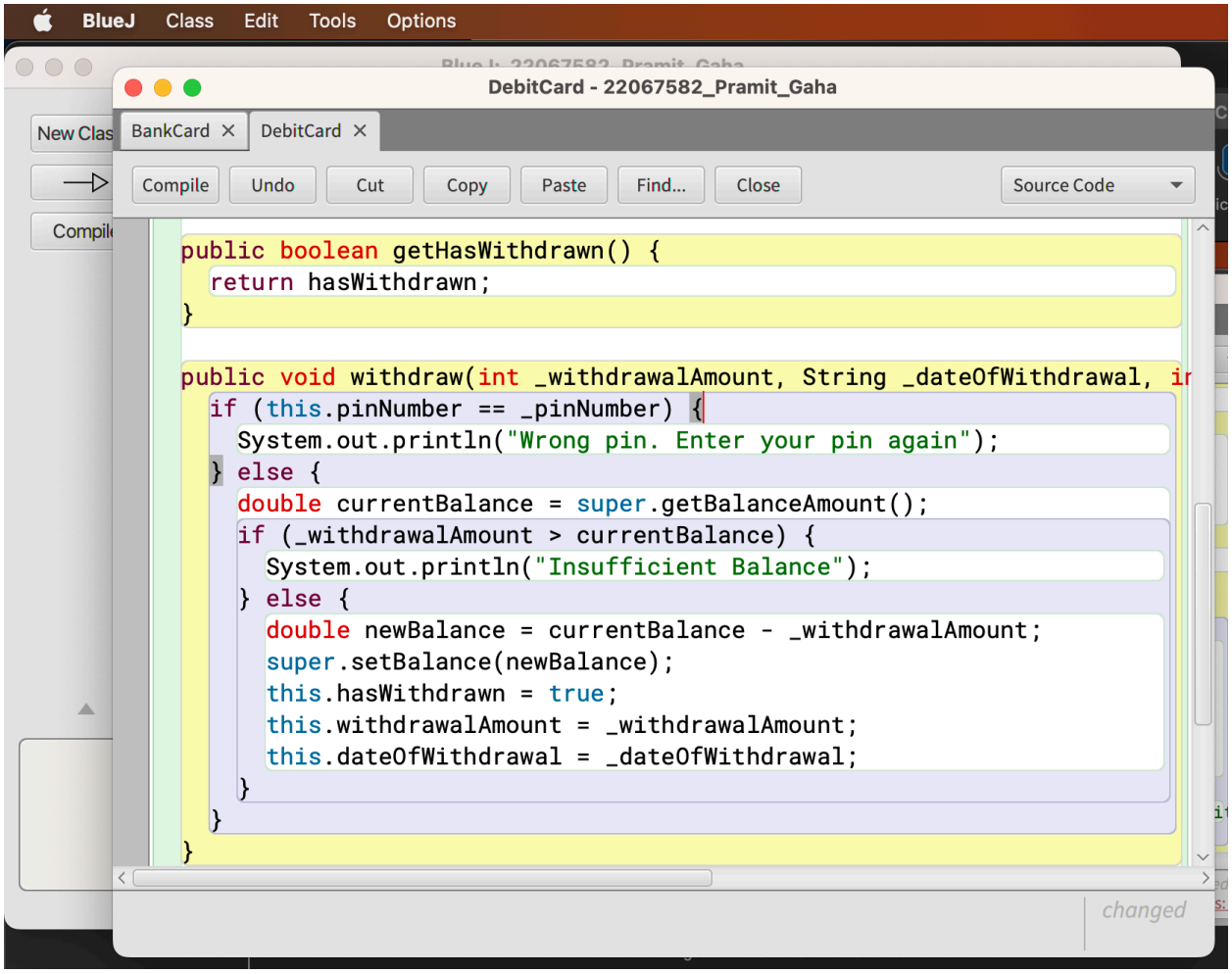


Figure 17. Screenshot for error 3

In the above shown screenshot, there is a logical error in the method withdraw. While checking at line this.pinNumber == _pinNumber. Instead of checking if the values aren't equal, values are equal check is done. So, when caller calls the with method with correct pin number still the method displays the message "Wrong pin. Enter your pin again". To fix the error inside the If statement this.pinNumber != pinNumber was added. Now it works in as it should have worked.

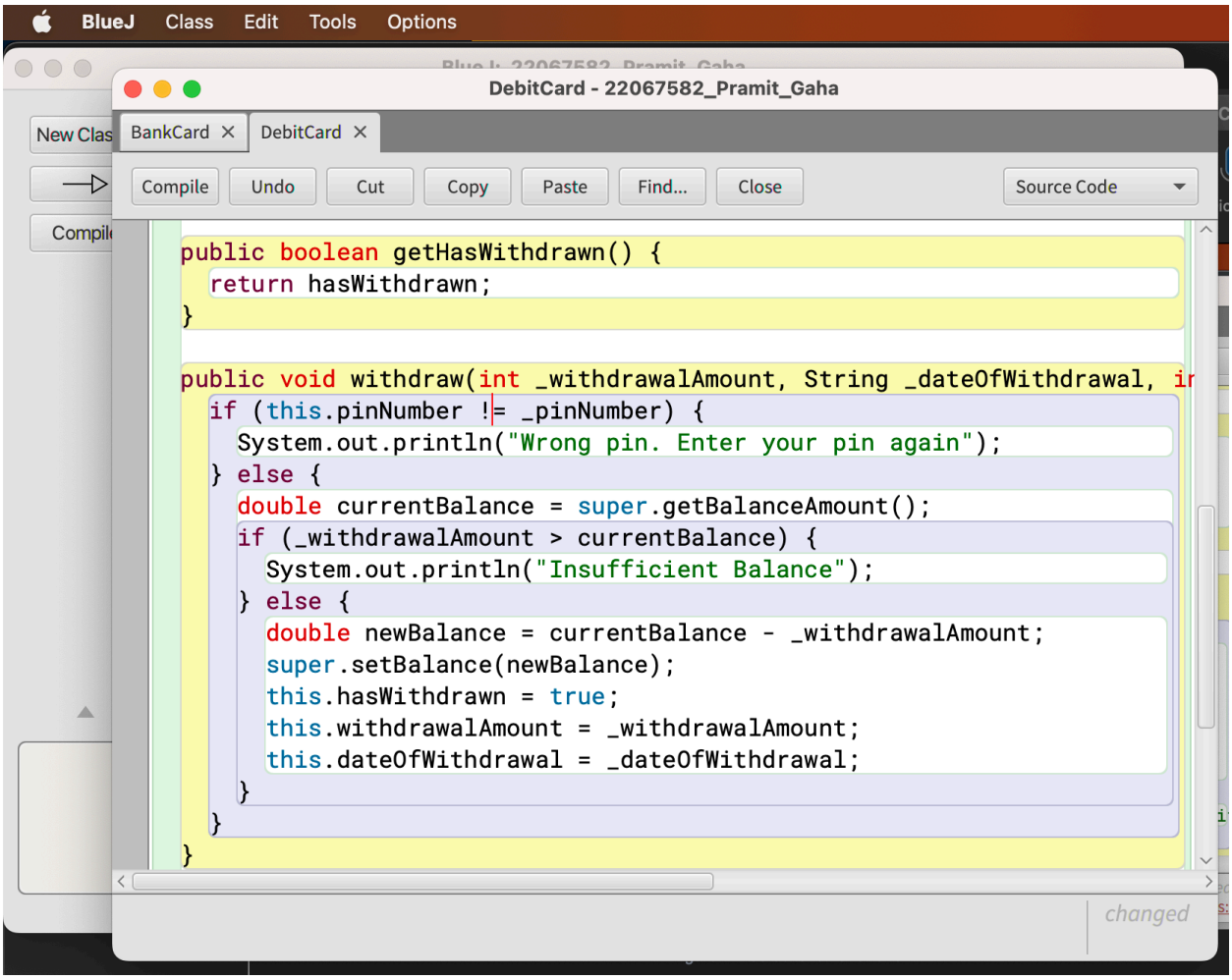


Figure 18. Screenshot for fixing error 3

7. Conclusion

The assignment was successfully finished ahead of the scheduled due date. The task involved extensive research as my prior knowledge of Java was limited. Initially, I encountered difficulties in creating a subclass and understanding how to interact with the methods of the parent class. However, through this coursework, I was able to overcome these challenges and acquire a deeper understanding of Java. Through various online resources and tutorials, I was able to resolve any errors that arose and gained a comprehensive understanding of the programming language. Additionally, I was able to grasp the fundamental concepts of object-oriented programming (OOP) in Java, which was a new and exciting learning experience for me. Overall, I found the coursework to be engaging and enjoyable.

8. References

Works Cited

GeeksForGeeks. (16 Jan, 2023). *Java Programming Language.* GeeksForGeeks.

JavaTpoint. (2021). *Java Tutorial.* JavaTpoint.

Mendeley. (2022). Mendeley.

(GeeksForGeeks, 16 Jan, 2023; JavaTpoint, 2021)

9. Appendix

9.1. BankCard.java

```
public class BankCard {

    private int cardId; // variable to store the cardId of the bank card

    private String clientName; // variable to store the name of the card holder

    private String issuerBank; // variable to store the name of the bank that issued the card

    private String bankAccount; // variable to store the bank account number of the card holder

    private double balanceAmount; // variable to store the current balance in the card


    /**
     * Constructor for objects of class BankCard
     * @param _balanceAmount the initial balance amount of the card
     * @param _cardId unique identification number of the card
     * @param _bankAccount the bank account associated with the card
     * @param _issuerBank the bank that issued the card
     */
    public BankCard(double _balanceAmount, int _cardId, String _bankAccount, String _issuerBank) {
        cardId = _cardId;
        clientName = "";
        issuerBank = _issuerBank;
        bankAccount = _bankAccount;
        balanceAmount = _balanceAmount;
    }


    /**
```

```
     * getter method for cardId variable
     * @return the cardId of the card
     */
    public int getCardId(){
        return cardId;
    }

    /**
     * getter method for clientName variable
     * @return the clientName of the card holder
     */
    public String getClientName(){
        return clientName;
    }

    /**
     * getter method for issuerBank variable
     * @return the bank that issued the card
     */
    public String getIssuerBank(){
        return issuerBank;
    }

    /**
     * getter method for bankAccount variable
     * @return the bank account associated with the card
     */
    public String getBankAccount(){
```

```
        return bankAccount;
    }

    /**
     * getter method for balanceAmount variable
     * @return the current balance amount of the card
     */
    public double getBalanceAmount(){
        return balanceAmount;
    }

    /**
     * setter method for clientName variable
     * @param _clientName the name of the card holder
     */
    public void setClientName(String _clientName){
        this.clientName = _clientName;
    }

    /**
     * setter method for balanceAmount variable
     * @param _balanceAmount the new balance amount of the card
     */
    public void setBalance(double _balanceAmount){
        this.balanceAmount = _balanceAmount;
    }

    /**
```

```
     * method to display the card details
     */
    public void display(){
        if (this.clientName.isEmpty()){
            System.out.println("Client Name isn't assigned. Please assign a Client
Name");
        }else{
            System.out.println("Card Id: " + this.cardId);
            System.out.println("Client Name: " + this.clientName);
            System.out.println("Issuer Bank: " + this.issuerBank);
            System.out.println("Bank Account: " + this.bankAccount);
            System.out.println("Balance Amount: " + this.balanceAmount);
        }
    }
}
```

9.2. DebitCard.java

```
public class DebitCard extends BankCard {
  private int pinNumber;
  private int withdrawalAmount;
  private String dateOfWithdrawal;
  private boolean hasWithdrawn;

  public DebitCard(double _balanceAmount, int _cardId, String _issuerBank,
String _bankAccount, String _clientName, int _pinNumber){
    super(_balanceAmount, _cardId, _bankAccount, _issuerBank);
    super.setClientName(_clientName);
    pinNumber = _pinNumber;
    hasWithdrawn = false;
  }

  public int getPinNumber() {
    return pinNumber;
  }

  public int getWithdrawalAmount() {
    return withdrawalAmount;
```

```
  }

  public String getDateOfWithdrawal() {
    return dateOfWithdrawal;
  }

  public boolean getHasWithdrawn() {
    return hasWithdrawn;
  }

  public void withdraw(int _withdrawalAmount, String _dateOfWithdrawal, int
_pinNumber) {
    if (this.pinNumber != _pinNumber) {
      System.out.println("Wrong pin. Enter your pin again");
    } else {
      double currentBalance = super.getBalanceAmount();
      if (_withdrawalAmount > currentBalance) {
        System.out.println("Insufficient Balance");
      } else {
        double newBalance = currentBalance - _withdrawalAmount;
        super.setBalance(newBalance);
        this.hasWithdrawn = true;
        this.withdrawalAmount = _withdrawalAmount;
        this.dateOfWithdrawal = _dateOfWithdrawal;
      }
    }
  }

  @Override
  public void display() {
    super.display();
    System.out.println("Pin Number: " + this.pinNumber);
    if (this.hasWithdrawn) {
      System.out.println("Withdrawal Amount: " + this.withdrawalAmount);
      System.out.println("Date of Withdrawal: " + this.dateOfWithdrawal);
    }
  }
}
```

9.3. CreditCard.java

```
public class CreditCard extends BankCard
{
   private int cvcNumber;
```

```
    private double creditLimit;
    private double interestRate;
    private String expirationDate;
    private int gracePeriod;
    private boolean isGranted;


    /**
     * Constructor for objects of class CreditCard
     */
    public CreditCard(int _cardId, String _clientName, String _issuerBank, String
_bankAccount, double _balanceAmount, int _cvcNumber, double _interestRate,
String _expirationDate)
    {
        super(_balanceAmount, _cardId, _bankAccount, _issuerBank);
        super.setClientName(_clientName);
        cvcNumber = _cvcNumber;
        interestRate = _interestRate;
        expirationDate = _expirationDate;
        isGranted = false;
    }

    public int getCvcNumber(){
        return cvcNumber;
    }
    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){
        double allowedCreditLimit = 2.5 * super.getBalanceAmount();
```

```
        if (_creditLimit <= allowedCreditLimit){
            this.creditLimit = _creditLimit;
            this.gracePeriod = _gracePeriod;
            this.isGranted = true;
        }else{
            System.out.println("Credit limit is higher than the allowed limit, Please
input a lower credit limit than " + allowedCreditLimit);
        }
    }

    public void cancelCreditCard(){
        this.cvcNumber = 0;
        this.creditLimit = 0;
        this.gracePeriod = 0;
        this.isGranted = false;
    }

    @Override
    public void display() {
        if (this.isGranted == true) {
            super.display();
            System.out.println("CVC Number: " + cvcNumber);
            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 has not been granted. Credit Limit and Grace
Period not available.");
        }
    }
}
```