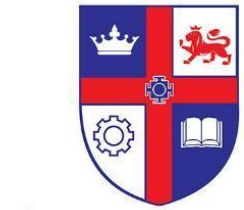
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2022-23 Autumn

Student Name: Sajin Raj Amatya

London Met ID: 22067177

College ID: NP01AI4A220003

Group: AI1

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 Content

List of Figures

List of Tables

1) Introduction

Java is an object-oriented, write-once, run-anywhere, high-level, general-purpose programming language created by Sun Microsystems in 1995 (Shankar, 2023). Java syntax is based on the C++ language where there is no need to remove the unrelated object since there is an automatic garbage collection in it. Once you have written the code in java, you can run it on any platform as it is a platform-independent or cross-platform programming language (Coursera, 2022).

Java support the object-oriented programming paradigm which is based on the idea of object where object consist the data in the form of attributes, Class is a group of different object which have same properties and common behaviour, it is also called as the blue print of the object and the method is the reusable block of code that carries certain task (Cyubahiro, 2022).

Here, in programming individual coursework we are assigned to implement a real-world problem scenario with the help of a java object-oriented concept by creating a parent or main class which signifies Bank card and two subclasses or child class Debit Card and Credit Card. Here the concept of inheritance and encapsulation are used where each attribute of the parent class and child class is set to private access modifier. In order to get an access to the attribute of a superclass or parent class getter and setter method are used.

The Subclasses Debit Card and Credit Card inherits the properties and behaviours of a parent class Bank Card. Bank card parent class store the detail of the client such as balance amount, card Id, bank account, and issuer bank and display it if required. Debit card child class is used for withdrawing the required amount and Credit card child class is used for setting the credit limit and cancelling the credit card according to the requirement of a client. The tools that are used in this assignment are Bluej software for java programming, Microsoft word for documentation section and Google scholar and other journal sites for references.

2) Class Diagram

Class Diagram is a type of diagram which is used to describe the structure of a program in the form of classes, attribute and methods in a pictorial manner, and it also defines the relation between them.

2.1) Class Diagram of Bankcard parent class

Bankcard
- balance_Amount: int - card_Id: int - issuer_Bank: String - client_Name: String - bank_Account : String
+<<constructor>>Bankcard (balance_Amount:int, card_Id: int, Issuer_Bank : String, bank_Account : String) + getBalance_Amount() : int + getCard_Id() : int + getIssuer_Bank () : String + getClient_Name (): String + getBank_Account () : String + setBalance_Amount (balance_Amount:int): void + setClient_Name(client_Name: String): void + display(): void

Table 1 Class Diagram of the Bankcard parent class

2.2) Class Diagram of Debitcard child class

Debitcard
- pin_Number: int - date_Of_Withdrawal: String - withdrawal_Amount: int - has_Withdrawn : boolean
+<<constructor>>Bankcard (balance_Amount : int , card_Id : int, bank_Account : String Issuer_Bank : String, client_Name:String , pin_Number:int) +getPin_Number () : int +getWithdrawal_Amount (): int + getDate_Of_Withdrawal () : String +getHas_Withdrawn (): boolean +setWithdrawal_Amount (withdrawal_Amount: int) : void +withdraw (withdrawal_Amount: int, pin_Number:int): void + display (): void

Table 2 Class Diagram of Debitcard child class

2.3) Class Diagram of Creditcard child class

Creditcard
- cvc_number: int - credit_Limit: double - interest_Rate: double - expiration_Date: String - grace_Period: int - is_Granted : boolean
+<<constructor>>Creditcard (card_Id : int, client_Name : String, issuer_Bank : String, bank_Account : String, balance_Amount: int, cvc_Number : int, interest_Rate : double, expiration_Date : String) + getCvc_Number (): int + getCredit_Limit (): double + getInterest_Rate (): double + getExpiration_Date (): String + getGrace_Period (): int + getIs_Granted (): Boolean + setCredit_Limit (new_Creditlimit : double, new_Gracelimit : int) : void + cancel_Credit_Card (): void + display() : void

Table 3 Class Diagram of Creditcard child class

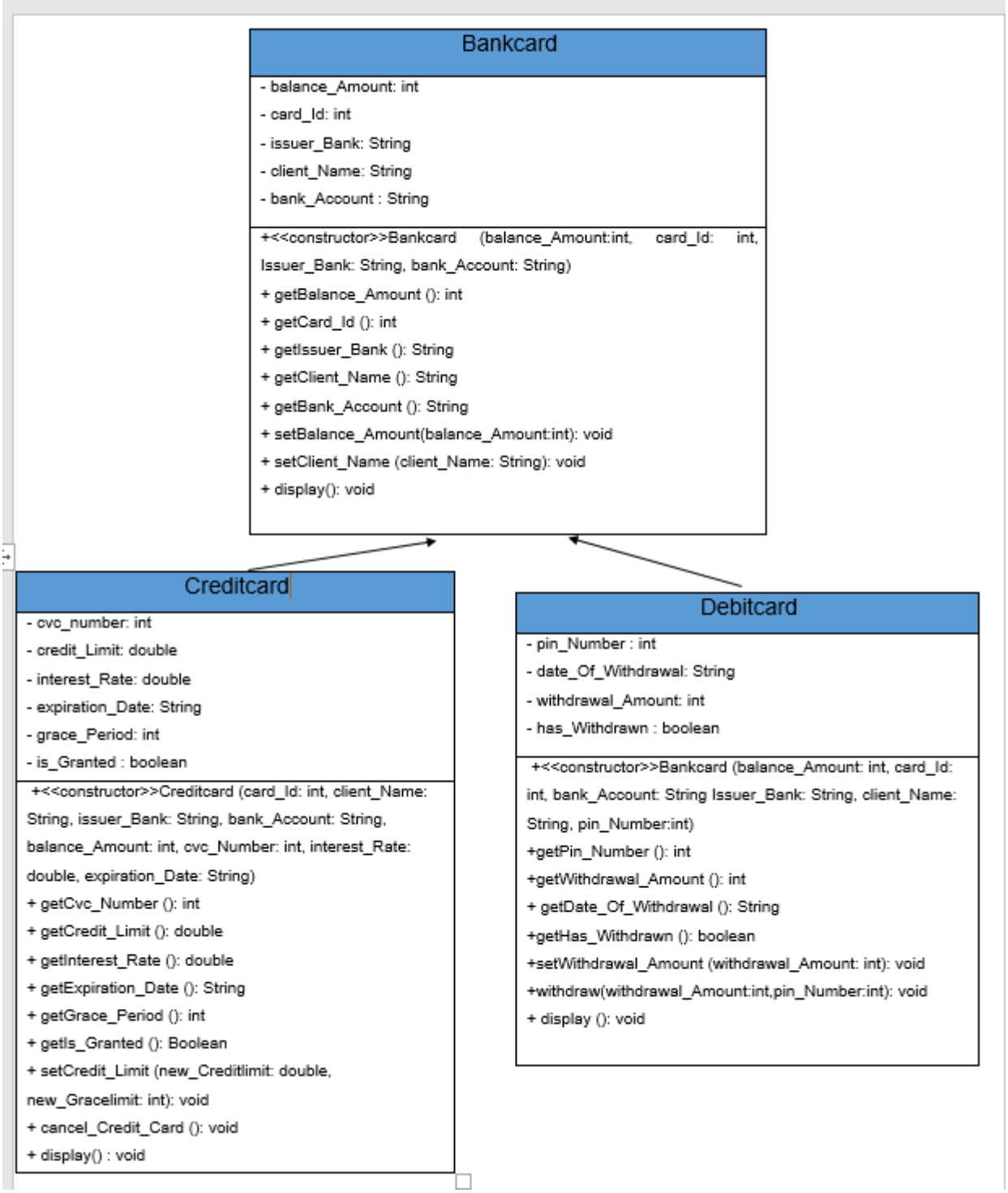


Figure 1 Inheritance diagram

3) Pseudocode

Pseudo code is the method to represent the steps of an algorithm in an understandable manner for non-programmer that ensures easy readability of a code for effectiveness interaction with different background people (Pepcoding, 2021).

3.1) Pseudocode for Bankcard class

CREATE a parent class Bankcard using public access modifier

DO

DECLARE instance variable balance_Amount as int using private access modifier

DECLARE instance variable card_Id as int using private access modifier

DECLARE instance variable issuer_Bank as String using private access modifier

DECLARE instance variable client_Name as String using private access modifier

DECLARE instance variable bank_Account as String using private accessmodifier

CREATE constructor of Bank card which accept four parameters in balance_Amount, int card_Id, String issuer_Bank, String bank_A

DO

ASSIGN balance_ Amount attribute to parameter balance_Amount

ASSIGN card_Id attribute to parameter card_Id

ASSIGN client_Name attribute to empty string

ASSIGN issuer_Bank attribute to parameter issuer_Bank

ASSIGN bank_Account attribute to parameter bank_Account

END DO

CREATE accessor method getBalance_Amount() with return type int

DO

RETURN balance_Amount

END DO

CREATE accessor method getCard_Id() with return type int

DO

RETURN card_Id

END DO

CREATE accessor method getIssuer_Bank() with return type String

DO

RETURN issuer_Bank

END DO

CREATE accessor method getClient_Name() with return type String

DO

RETURN client_Name

END DO

CREATE accessor method getBank_Account() with return type String

DO

RETURN bank_Account

END DO

CREATE mutator method setBalance_Amount() with parameter int balance_Amount

DO

ASSIGN balance_Amount attribute to value of parameter balance_Amount

END DO

CREATE mutator method setClient_Name() with parameter String client_Name

DO

ASSIGN client_Name attribute to value of parameter client_Name

END DO

CREATE a method display() with no return type

DO

PRINT Card ID

IF client_Name is empty **THEN**

PRINT Please enter the client name

ELSE

PRINT ClientName

END IF

PRINT IssuerBank

PRINT BankAccount

PRINT BalanceAmount

END DO

3.2) Pseudocode for Debitcard class

CREATE a child class Debitcard using public access modifier

DO

DECLARE instance variable pin_Number as int using private access modifier

DECLARE instance variable date_Of_Withdrawal as String using private access modifier

DECLARE instance variable withdrawal_Amount as int using private access modifier

DECLARE instance variable has_Withdrawn as boolean using private access modifier

CREATE constructor of Debitcard with a parameter int balance_Amount, int card_Id, String bank_Account,String issuer_Bank, String client_Name, int pin_Number

DO

CALL constructor of parent class with parameter balance_Amount, card_Id, issuer_Bank, bank_Account

CALL superclass mutator method setClient_Name(client_Name)

ASSIGN pin_Number attribute to value of parameter pin_Number

ASSIGN has_Withdrawn attribute to false

END DO

CREATE an accessor method getPin_Number () with return type int

DO

RETURN pin_Number

END DO

CREATE an accessor method getWithdrawal_Amount() with return type int

DO

RETURN withdrawal_Amount

END DO

CREATE an accessor method getDate_Of_Withdrawal() with return type String

DO

RETURN date_Of_Withdrawal

END DO

CREATE an accessor method getHas_Withdrawn() with return type boolean

DO

RETURN has_Withdrawn

END DO

CREATE a mutator method setWithdrawal_Amount() with parameter int withdrawal_Amount

DO

ASSIGN withdrawal_Amount attribute to value of parameter withdrawal_Amount

END DO

CREATE a method withdraw with parameter int withdrawal_Amount,String, date_Of_Withdrawal, int pin_Number

DO

IF pin_Number attribute is equal to the value of parameter pin_Number **THEN**

IF withdrawal_Amount is less or equal to getBalance_Amount() **THEN**

SET balance_Amount equal to subtract balance_Amount to withdrawal_Amount

SET has_Withdrawn attribute to true

END IF

ELSE

PRINT Not enough balance in your account

END IF

ELSE

PRINT Incorrect Pin number, please try again

END ELSE

END DO

CREATE a method display() with no return type

DO

CALL Display method of superclass

IF has_Withdrawn is set equal to true **THEN**

PRINT Pin number

PRINT Withdrawal Amount

PRINT Date of withdrawal

END IF

` **ELSE**

PRINT Withdrawal transaction has not been carried out yet!

PRINT Balance amount

END ELSE

END DO

3.3) Pseudocode for Creditcard class

CREATE a child class Creditcard using public accessor modifier

DO

DECLARE instance variable cvc_Number as int using private access modifier

DECLARE instance variable credit_Limit as double using private access modifier

DECLARE instance variable interest_Rate as double using private access modifier

DECLARE instance variable expiration_Date as String using private access modifier

DECLARE instance variable grace_Period as int using private access modifier

DECLARE instance variable is_Granted as boolean using private access modifier

CREATE a constructor of Creditcard with parameter int card_Id, String client_Name, String issuer_Bank, String bank_Account, int balance_Amount, int cvc_Number,double interest_Rate, String expiration_Date.

DO

CALL constructor of parent class with parameter balance_Amount, card_Id, issuer_Bank, bank_Account

ASSIGN cvc_Number attribute to value of parameter cvc_Number

ASSIGN interest_Rate attribute to value of parameter interest_Rate

ASSIGN expiration_Date attribute to value of parameter expiration_Date

SET is_Granted attribute to false

CALL method setBalance_Amount and SET the value of balance_Amount attribute

CALL method setClient_Name and SET the value of a client_Name attribute

END DO

CREATE an accessor method getCvc_Number() with return type int

DO

RETURN cvc_Number

END DO

CREATE an accessor method getCredit_Limit() with return type double

DO

RETURN credit_Limit

END DO

CREATE an accessor method getInterest_Rate() with return type double

DO

RETURN interest_Rate

END DO

CREATE an accessor method getExpiration_Date() with return type String

DO

RETURN expiration_Date

END DO

CREATE an accessor method getGrace_Period() with return type int

DO

RETURN grace_Period

END DO

CREATE an accessor method getIs_Granted() with return type boolean

DO

RETURN is_Granted

END DO

CREATE a method setCredit_Limit with parameter “doublenew_Creditlimit”, “int new_Gracelimit”

DO

IF credit_Limit is less or equal to two times the balance_Amount **THEN**

SET is_Granted to true

ASSIGN credit_Limit attribute to new_Creditlimit

END IF

ELSE

PRINT the credit amount can’t be issued, please try again later

END DO

END ELSE

CREATE a method cancel_Credit_Card with no return type

DO

SET credit_Limit attribute to zero

SET cvc_Number attribute to zero

SET grace_Period attribute to zero

SET is_Granted attribute to false

END DO

CREATE a method display with no return type

DO

CALL Display method of superclass

Print CVC Number

Print Interest Rate

Print Expiration Date

IF is_Granted attribute is equal to true **THEN**

Print Credit Limit

Print Grace Period

END IF

ELSE

Print Could not display Credit Limit and Grace period as it is not permitted

END ELSE

END DO

4) Method Description

Generally, in programming a method is a block of code which is reusable that carries out a certain task. A method is used to state the behaviours of the objects, and defines the action that its object can perform (Himanshi, 2022).

4.1) Bankcard

getBalance_Amount ()

It is a getter/accessor method with a return type integer which is used to return the value of instance variable balance_Amount when it is called.

getCard_Id ()

It is a getter/accessor method with a return type integer which is used to return the value of instance variable card_Id when it is called.

getIssuer_Bank ()

It is a getter/accessor method with a return type String which is used to return the value of instance variable issuer_Bank when it is called.

getClient_Name ()

It is a getter/accessor method with a return type String which is used to return the value of instance variable client_Name when it is called.

getBank_Account ()

It is a getter/accessor method with a return type String which is used to return the value of instance variable bank_Account when it is called.

setBalance_Amount ()

It is a mutator/setter method with a parameter “balance_Amount” which is used for assigning the value to instance variable balance_Amount .

setClient_Name ()

It is a mutator/setter method with a parameter “client_Name” which is used for assigning the value to instance variable client_Name.

void display ()

It is a method with no return type which is used for displaying the detail of a Bankcard class.

4.2) Debitcard

getPin_Number ()

It is a getter/accessor method with a return type integer which is used to return the value of instance variable pin_Number when it is called.

getWithdrawal_Amount ()

It is a getter/accessor method with a return type integer which is used to return the value of instance variable withdrawal_Amount when it is called.

getDate_Of_Withdrawal ()

It is a getter/accessor method with a return type String which is used to return the value of instance variable date_Of_Withdrawal when it is called.

getHas_Withdrawn ()

It is a getter/accessor method with a return type boolean which is used to return the value of instance variable has_Withdrawn when it is called.

setWithdrawal_Amount ()

It is a mutator/setter method with a parameter “withdrawal_Amount” which is used for assigning the value to instance variable withdrawal_Amount.

void withdraw ()

It is method with no return type which is used for withdrawing the amount from balance amount.

void display ()

It is a method with no return type which is used for displaying the detail of a debit card.

4.3) Creditcard

getCvc_Number ()

It is a getter/accessor method with a return type integer which is used to return the value of instance variable cvc_Number when it is called.

getCredit_Limit ()

It is a getter/accessor method with a return type double which is used to return the value of instance variable credit_Limit when it is called.

getInterest_Rate ()

It is a getter/accessor method with a return type double which is used to return the value of instance variable interest_Rate when it is called.

getExpiration_Date ()

It is a getter/accessor method with a return type String which is used to return the value of instance variable expiration_Date when it is called.

getGrace_Period ()

It is a getter/accessor method with a return type integer which is used to return the value of instance variable grace_Period when it is called.

getIs_Granted()
It is a getter/accessor method with a return type boolean which is used to return the value of instance variable is_Granted when it is called.

setCredit_Limit ()

It is a mutator/setter method with no return type and two parameters “double new_Creditlimit”, “int new_Grace” which is used to set the credit limit.

void cancel_Credit_Card ()

It is a method with no return type which is used to cancel the credit card of a client.

void display ()

It is a method with not return type which is used to display the detail of credit card.

5) Testing

Testing is the process of examining the program with the purpose to discover whether program fulfil the expected result successfully or not.

5.1) Test 1: 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 Debit Card Class
Action:	➢ The Debit card is called with the following arguments: balance_Amount = 7000 card_Id = 9871234 bank_Account = “Current” issuer_Bank = “Siddhartha Bank” client_Name = “Sajin Raj Amatya” pin_Number = 7879 ➢ Inspection of the Debit card class. ➢ void withdraw is called with the following argument. withdrawal_Amount = 5000 date_Of_Withdrawal = “2023-01-21” pin_Number = 7879 ➢ Re-inspection of the Debi card class
Expected Result:	The required amount would be withdrawn from the Debit card
Actual Result:	The required amount was withdrawn from the Debit card
Conclusion:	The test is successful.

Table 4 To Inspect the Debit Card class, withdraw the amount, and re-inspect

Output Result:

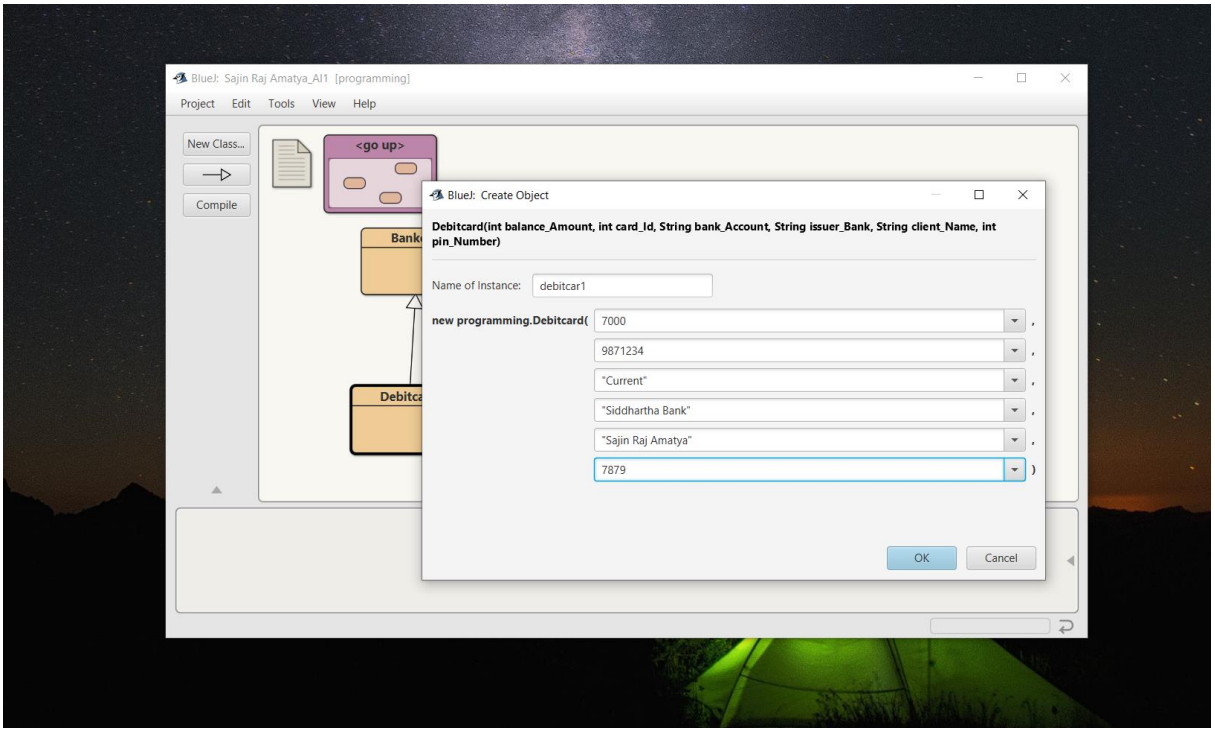


Figure 2 : Screenshot of creating object and assigning the data in Debit card class

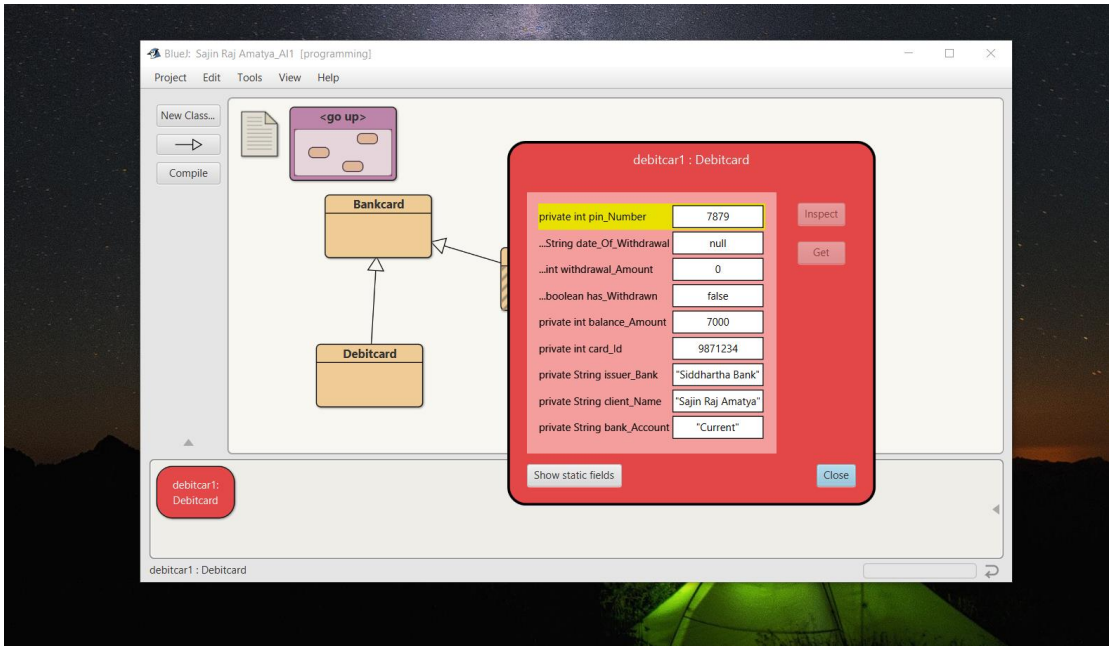


Figure 3 Screenshot for inspection of Debit card class

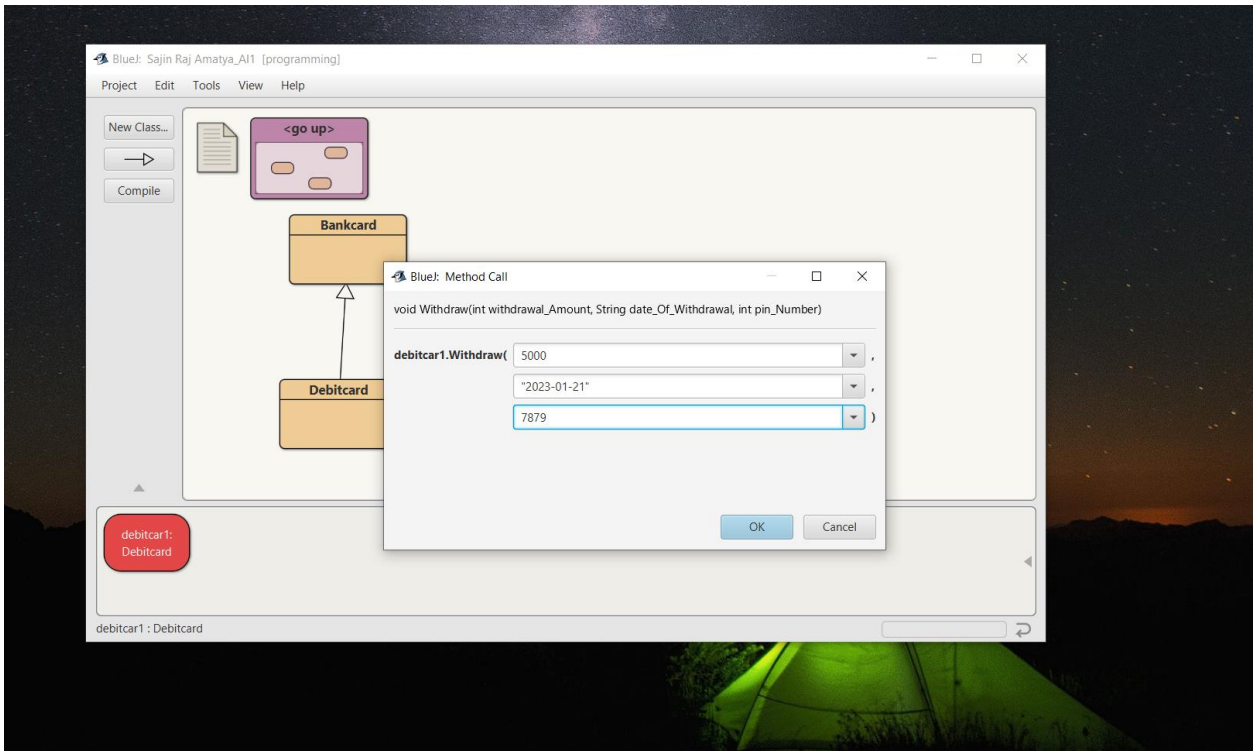


Figure 4 Screenshot for Inserting the data in void Withdraw method

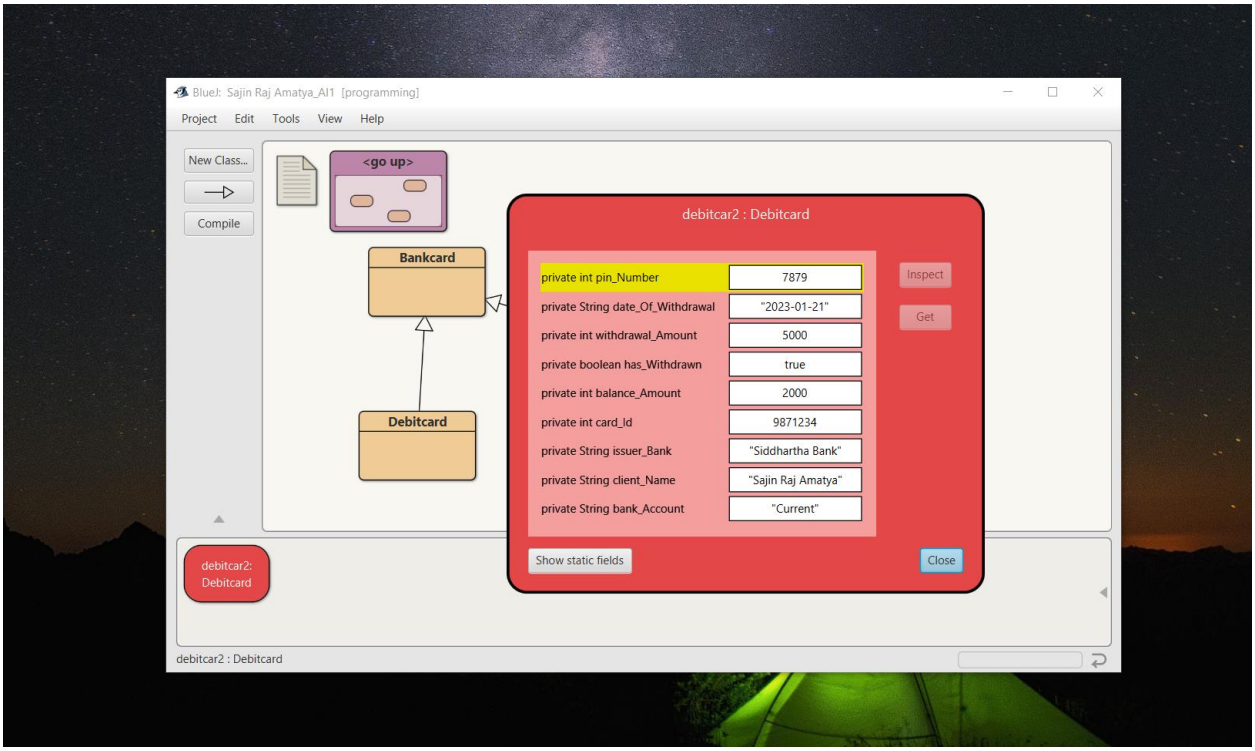


Figure 5 Screenshot for Re-inspection after withdrawing the amount

5.2) Test 2: Inspect Credit Card class, set the credit limit and reinspect the Credit Card class

Test No.	2
Objective:	To inspect the Credit Card class, set the credit limit and re-inspect the Credit Card class
Action:	➢ The Credit card is called with the following arguments: card_Id= 7879 client_Name = “Sajin Raj Amatya” issuer_Bank = “Siddhartha Bank” bank_Account = “Current” balance_Amount = 7000 cvc_Number = 334422 interest_Rate = 6.43 expiration_Date = “2023-09-14” ➢ Inspection of the Credit Card class. ➢ void setCredit_Limit is called with the following argument. new_Creditlimit = 20000 new_Graceperiod = 11 ➢ Re-inspection of the Credit card class
Expected Result:	The credit limit would be set on the Credit card
Actual Result:	The credit limit was set on the Credit card.
Conclusion:	The test is successful.

Table 5 To Inspect Credit Card class, set the credit limit and reinspect the

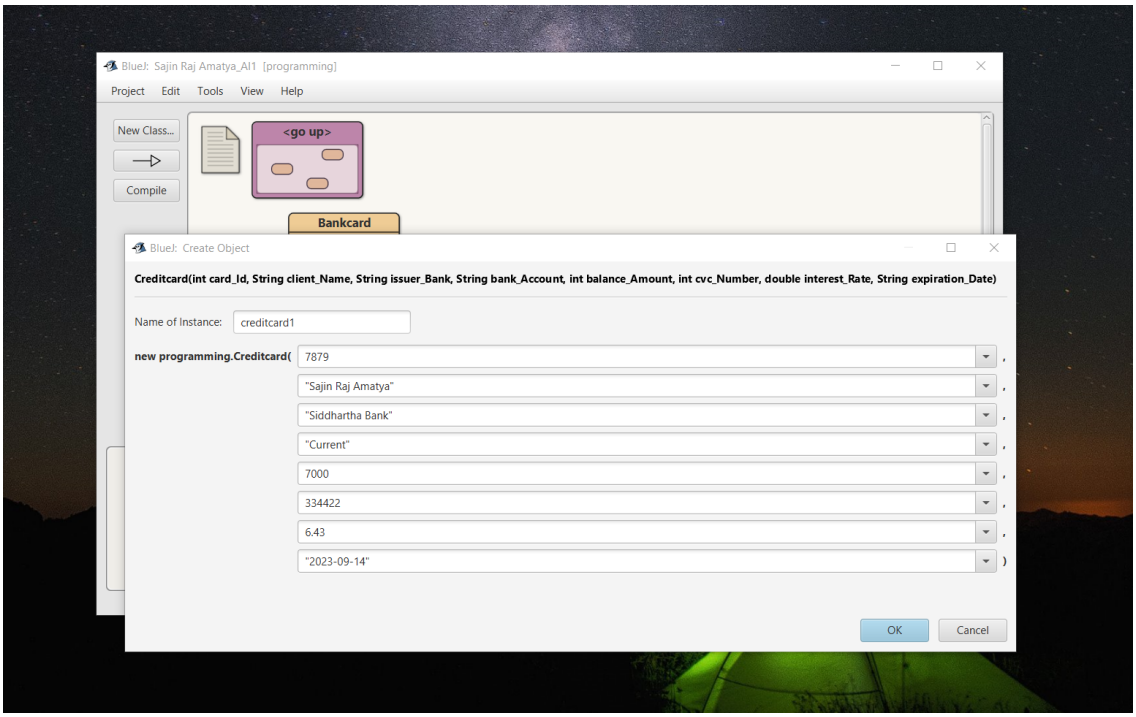


Figure 6 Screenshot of creating object and assigning the data in Credit card class

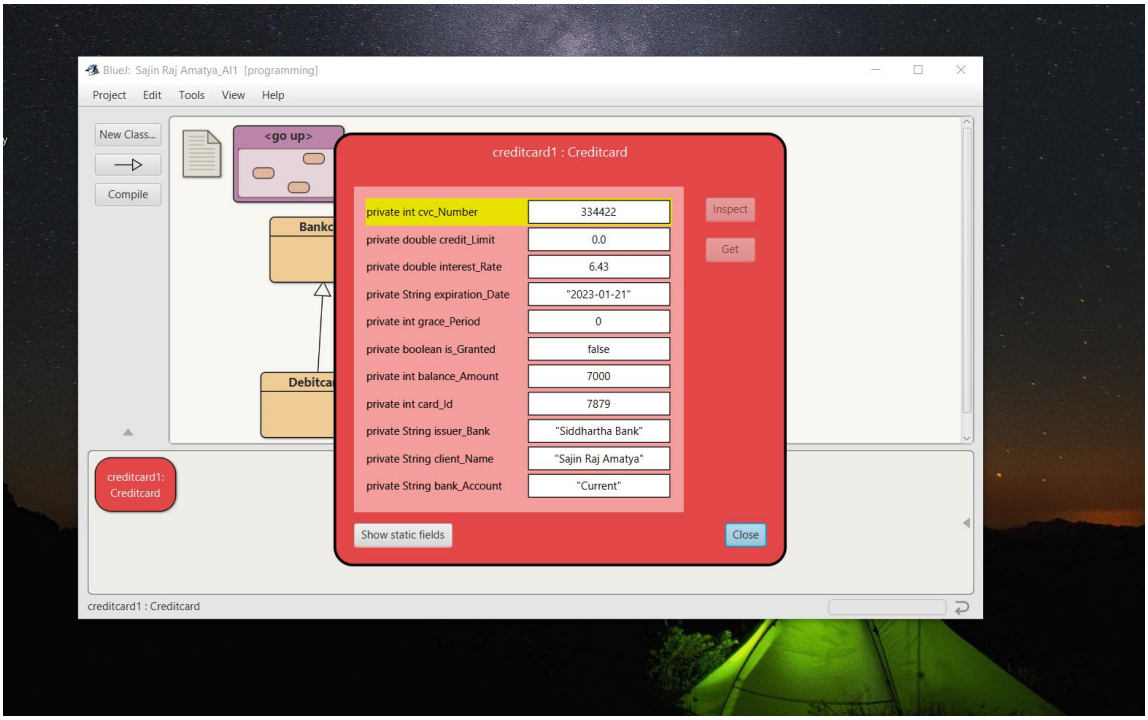


Figure 7 Screenshot for inspection of Credit card class

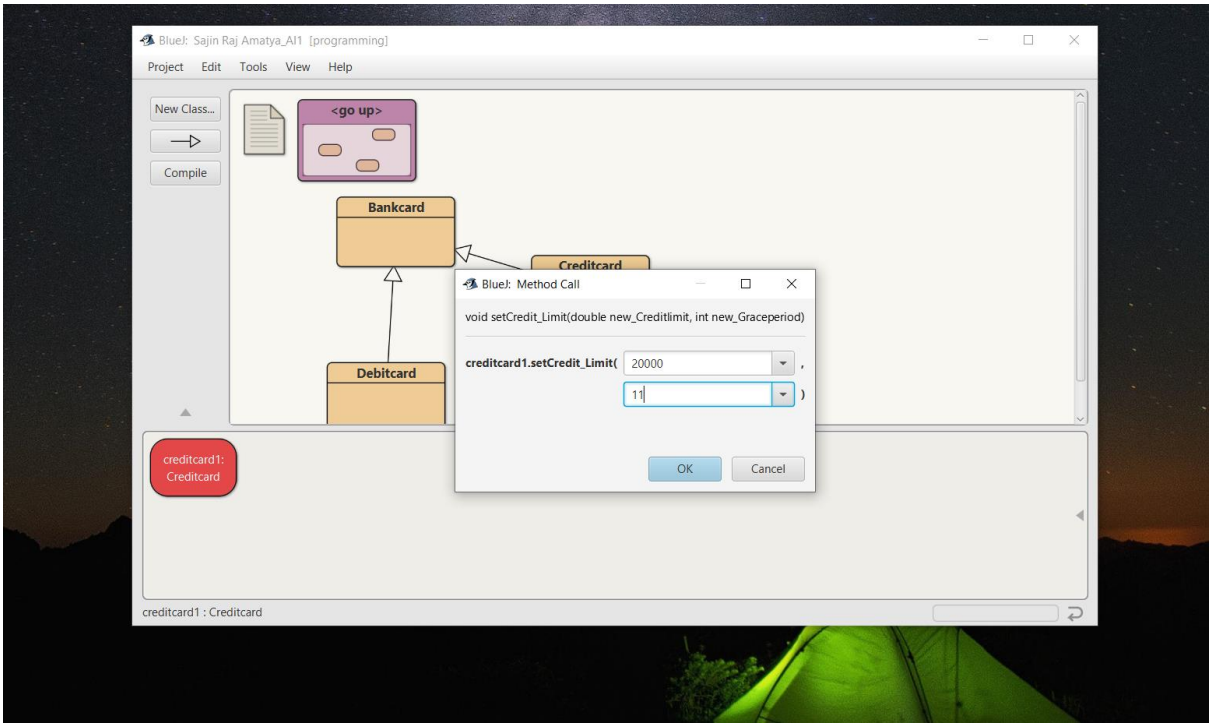


Figure 8 Screenshot of inserting data in void setCredit_Limit

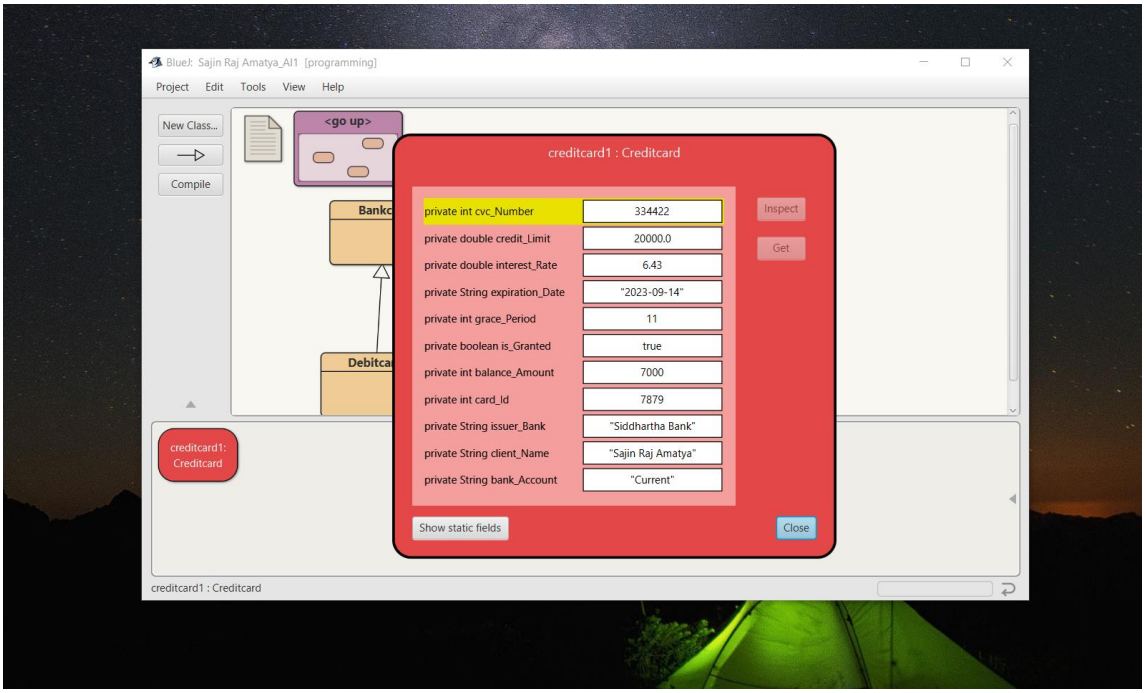


Figure 9 Screenshot of Re-inspection after setting credit limit

5.3) Test 3: Inspect Credit Card class again after cancelling the credit card.

Test No.	3
Objective:	To Inspect the Credit Card class again after canceling the credit card.
Action:	➢ The Credit card is called with the following arguments: card_Id= 7879 client_Name = “Sajin Raj Amatya” issuer_Bank = “Siddhartha Bank” bank_Account = “Current” balance_Amount = 7000 cvc_Number = 334422 interest_Rate = 6.43 expiration_Date = “2023-09-14” ➢ void setCredit_Limit is called with the following argument. new_Creditlimit = 20000 new_Graceperiod = 11 ➢ Inspection of a Credit before cancelling. ➢ The void cancelCredit_Card method is called where, Cvc_Number is set to zero. Credit_Limit is set to zero. Grace_Period is set to zero. Is_Granted is set to false. ➢ Inspection of a Credit card after cancelling.
Expected Result:	The detail of the Credit card would be removed
Actual Result:	The detail of the Credit card was removed
Conclusion:	The test is successful.

Table 6 To Inspect Credit Card class again after cancelling the credit card.

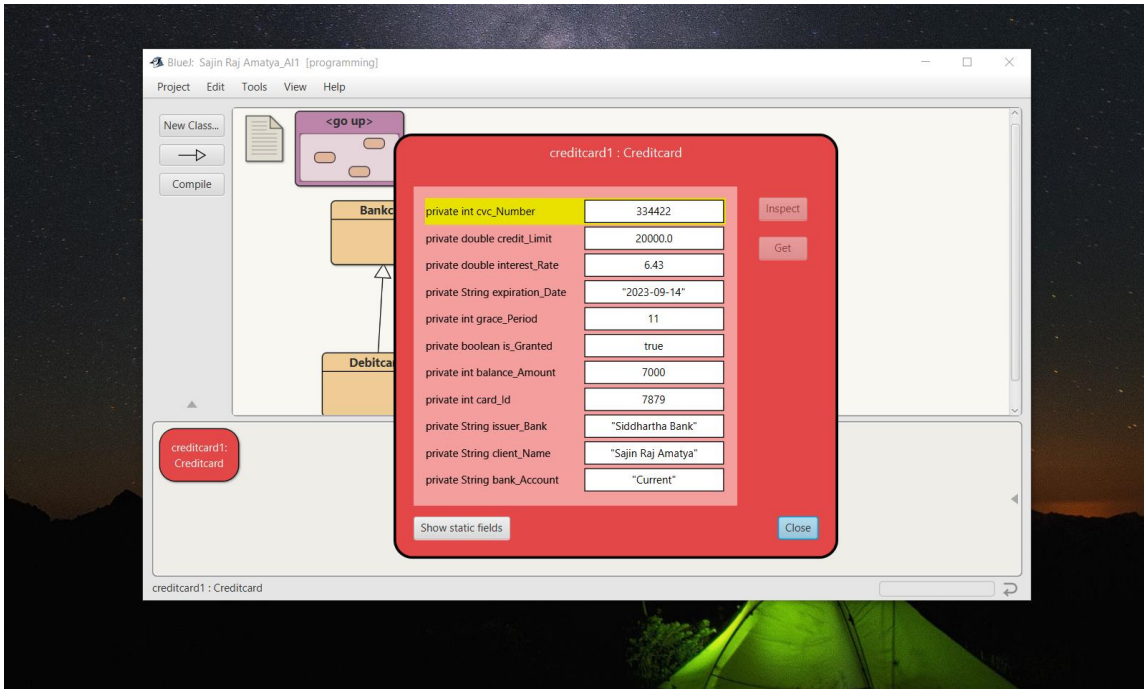


Figure 10 Screenshot for inspection of Credit card before canceling

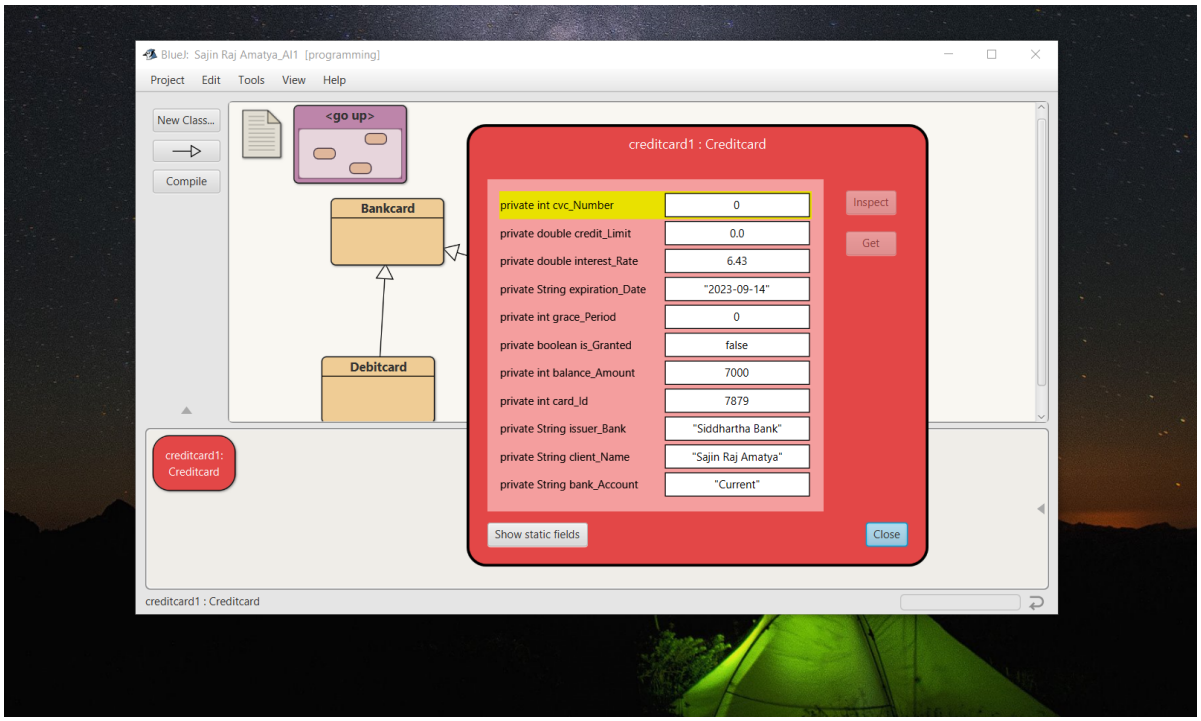


Figure 11 Screenshot of inspection after cancelling of Credit card

5.4) Test 4: Display the details of Debit Card and Credit Card classes.

Test No.	4
Objective:	To display the details of Debit Card and Credit Card classes.
Action:	➢ The Debit card is called with the following argument card_Id client_Name issuer_Bank bank_Account pin_Number withdrawal_Amount date_Of_Withdrawal ➢ The void display method is called from the Debit card class. ➢ Display the output of the Debit card class. ➢ The Credit card is called with the following argument card_Id client_Name issuer_Bank bank_Account pin_Number cvc_Number interest_Rate expiration_Date credit_Limit grace_Period ➢ The void display method is called from the Credit card class. ➢ Display the output of the Credit card class.
Expected Result:	The detail of the Credit card and Debit card would be displayed
Actual Result:	The detail of the Credit card and Debit card was displayed
Conclusion:	The test is successful.

Table 7 To Display the details of Debit Card and Credit Card classes.

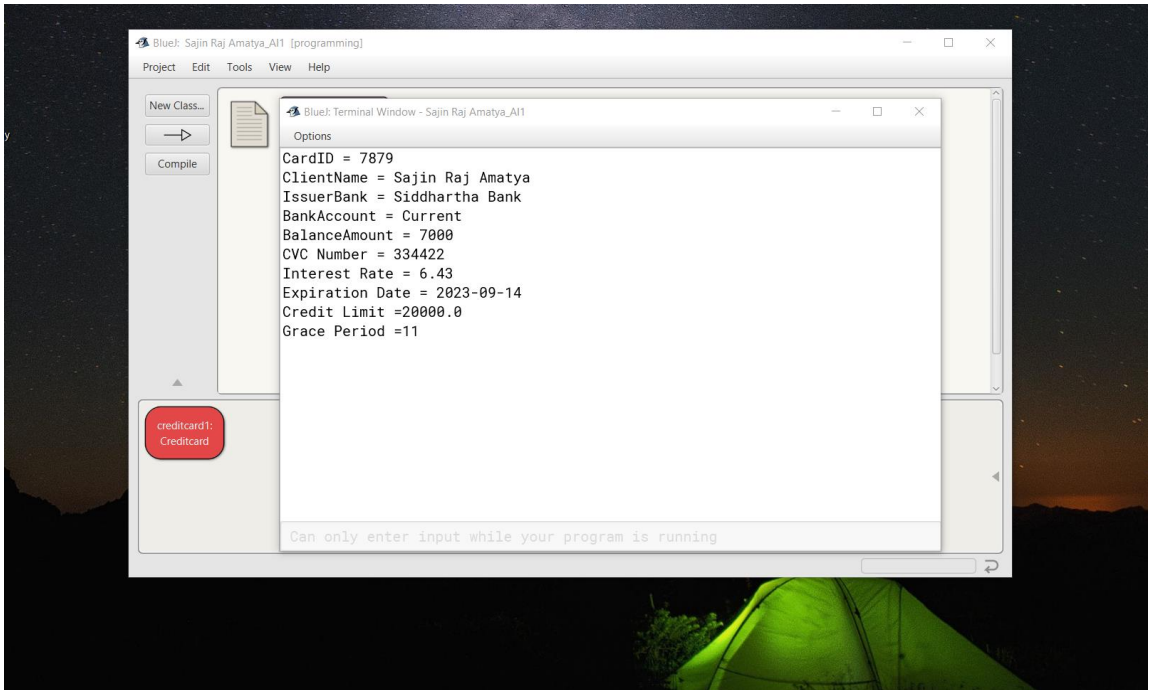


Figure 12 Screenshot for displaying the detail of Credit card class

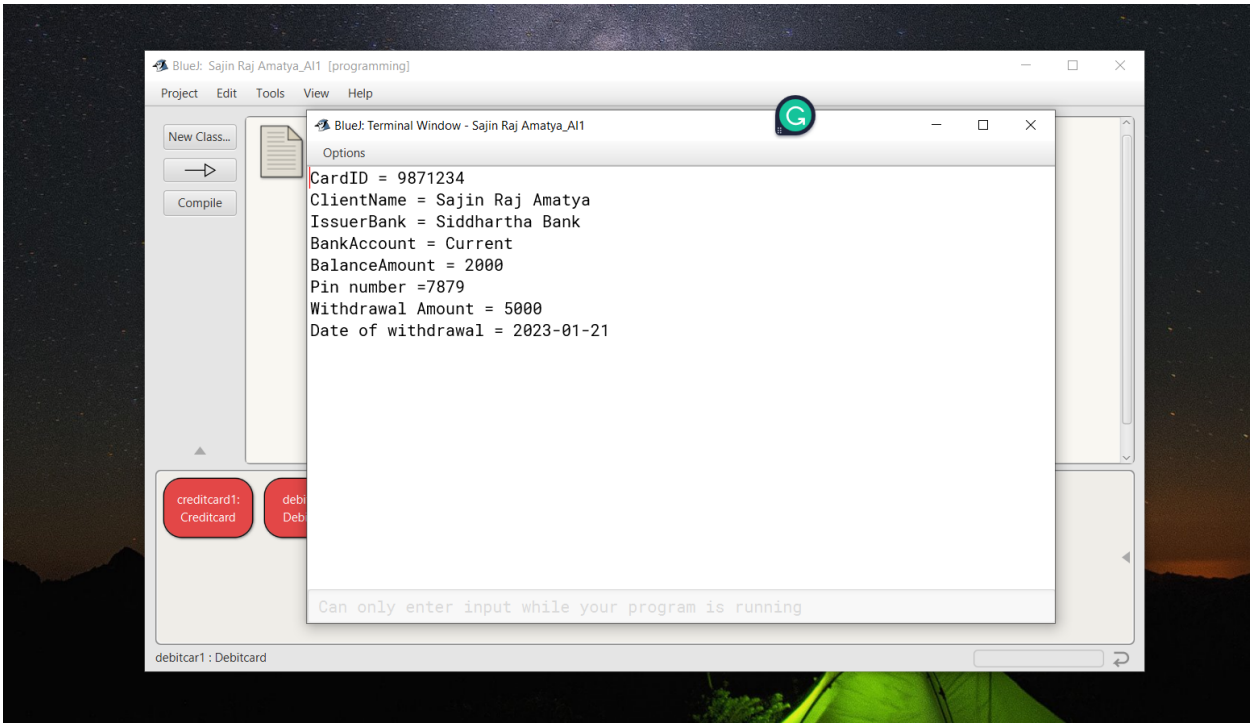


Figure 13 Screenshot for displaying the detail of a Debit card class

6) Error Detection and Correction

6.1) Syntax Error

The Syntax Error in programming is identified by the complier when there is a mistake in the development of a program such as mistyping keyword, no semicolon in the end of a line, improper labels, neglecting useful punctuation (Liang, 2015).

Error No.	1
Error type	Syntax error
Problem	After declaring the instance variable in the Bank card class compiler shows the error message which was shown due to the missing semicolon at the end of the line.
Solution	The semi-colon was added at the end of the line after declaring the instance variable.
Conclusion	The syntax error was detected and the correction was made to the code.

Table 8 Syntax Error detection and correction

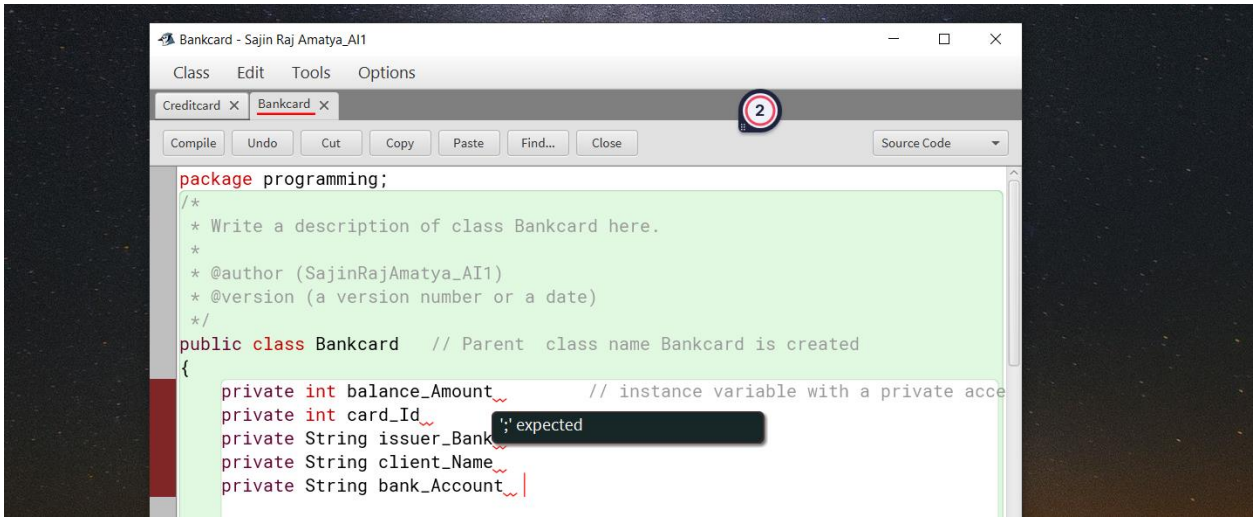


Figure 14 Screenshot of Syntax Error in the program

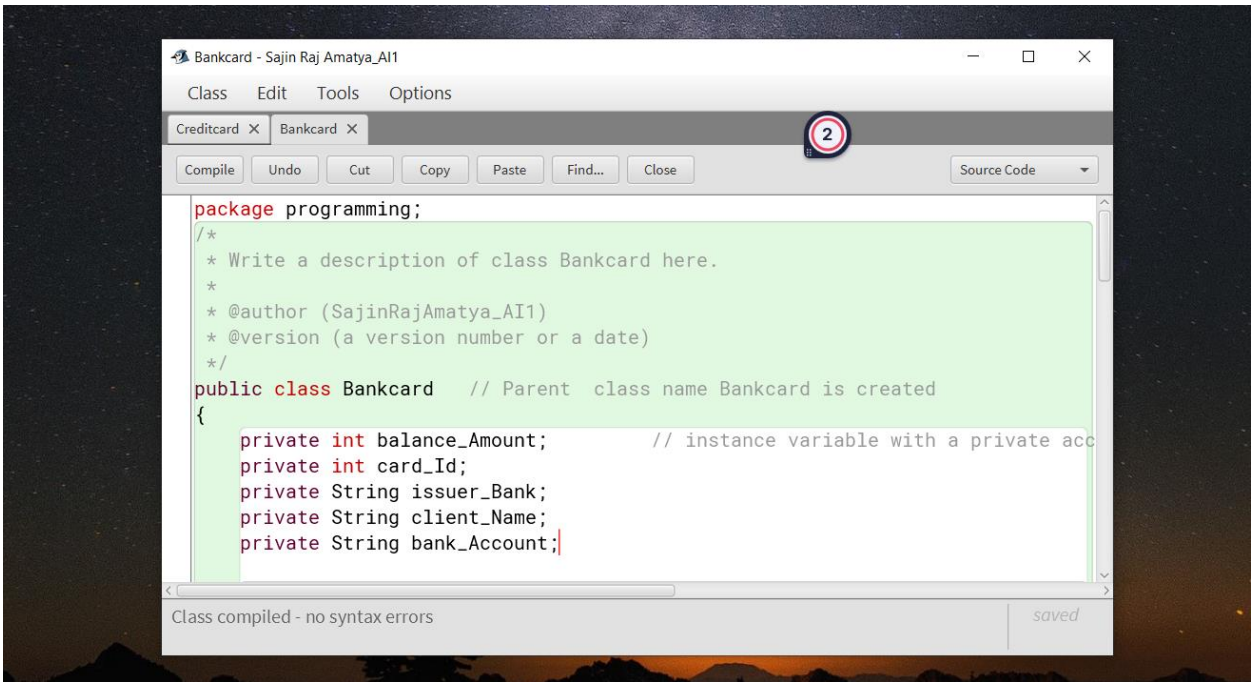


Figure 15 Screenshot of Correction of Syntax Error.

6.2) Logical error

In programming the logical error is a type of error which occurs when a program fails to perform specific task as per the requirement (Liang, 2015). It is not a mistake of a programming language rather it is a mistake for reasoning thinking of a programmer.

Error No.	2
Error type	Logical error
Problem	In the method setCredit_Limit if the credit limit is smaller or equal to the 2.5 times balance amount then the detail the of Credit limit and the Grace period is not shown. The Credit limit = 10000 Balance amount = 5000
Solution	The is_Granted attribute was set to true when the above condition is satisfied then the credit limit and Grace period would display.
Conclusion	The logical error was detected and the correction was made to the code.

Table 9 Logical Error detection and correction

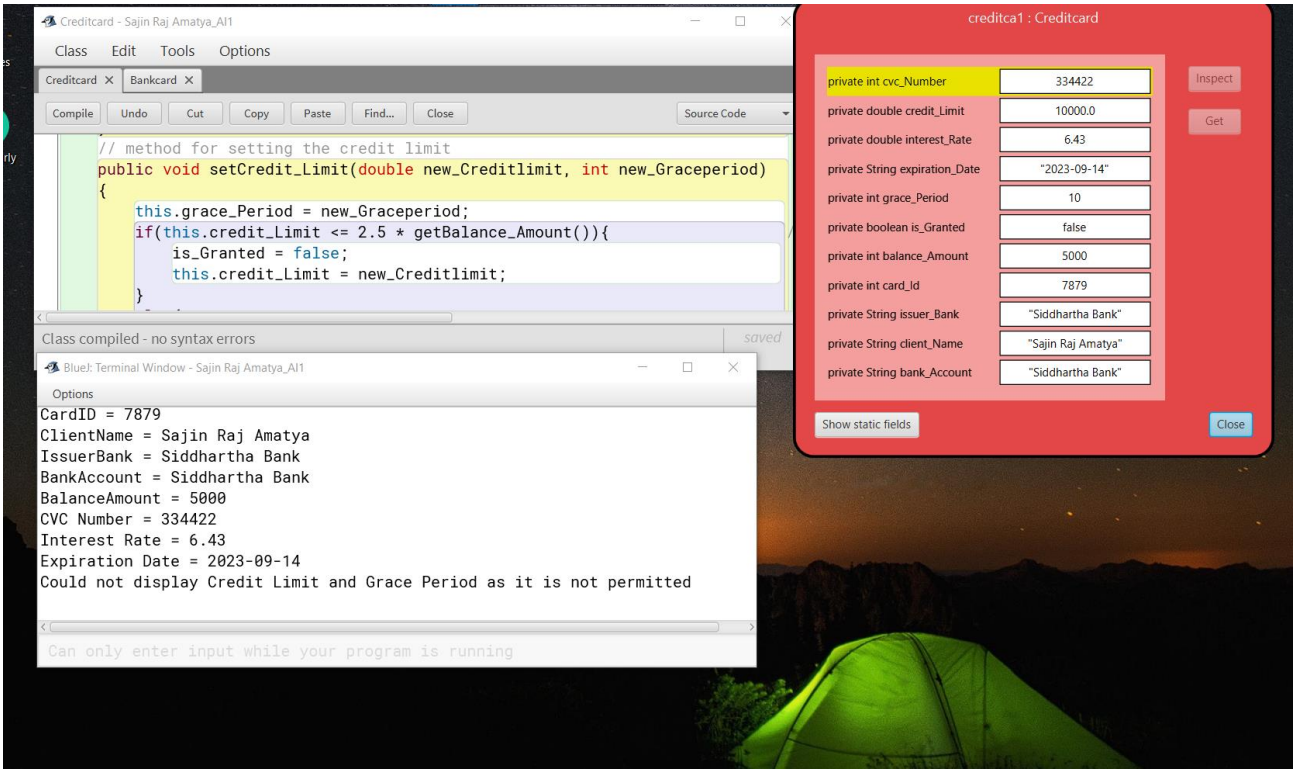


Figure 16 Screenshot of Logical Error in the program

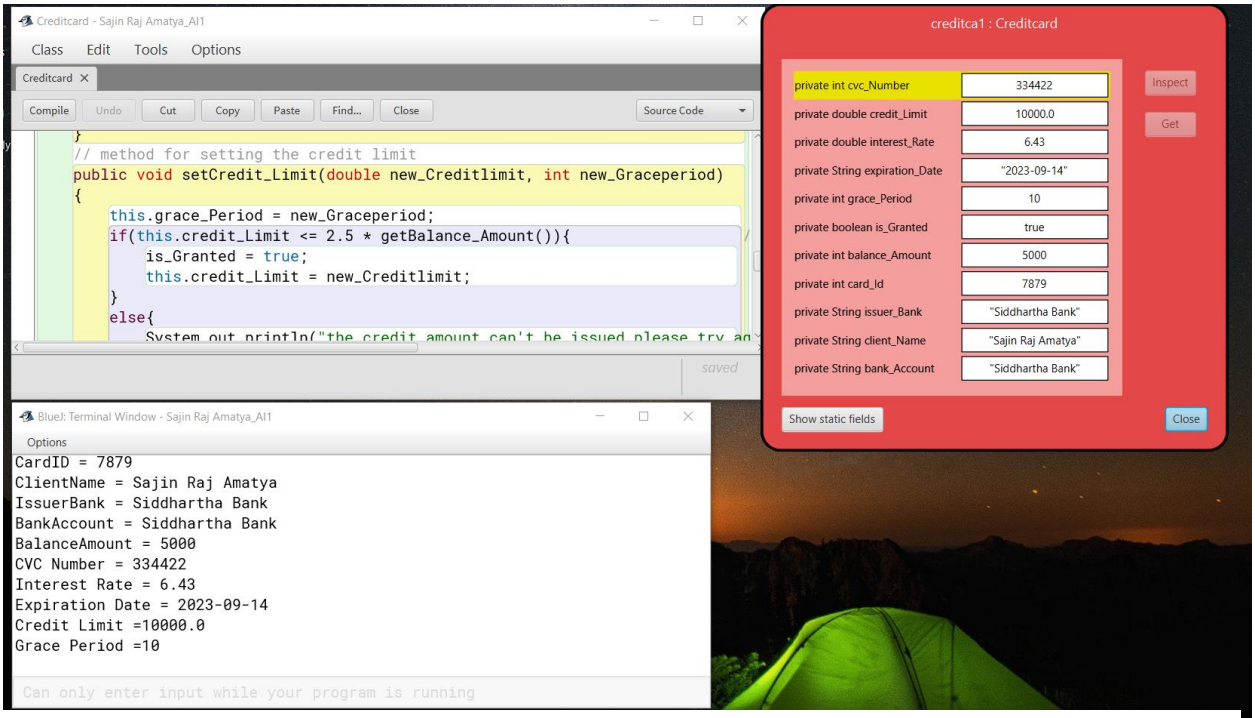


Figure 17 Screenshot of Correction of Logical Error

6.3) Semantic error

In programming semantic error is a type of error which have correct syntax but fails to perform the desired task due to the mistake in the meaning and context of a code. for example, using wrong datatype for variables, using undefined variable and wrong operator in a code.

Error No.	3
Error type	Semantic error
Problem	Access to the balance amount attribute of the parent class Bank card was not allowed to the child class Debit card while compiling the program.
Solution	The getter/accessor method is to be used in order to access the balance amount attribute with the private access modifier of parent class Bank card in Debit card child class
Conclusion	The semantic error was detected and the correction was made to the code.

Table 10 Semantic Error detection and correction

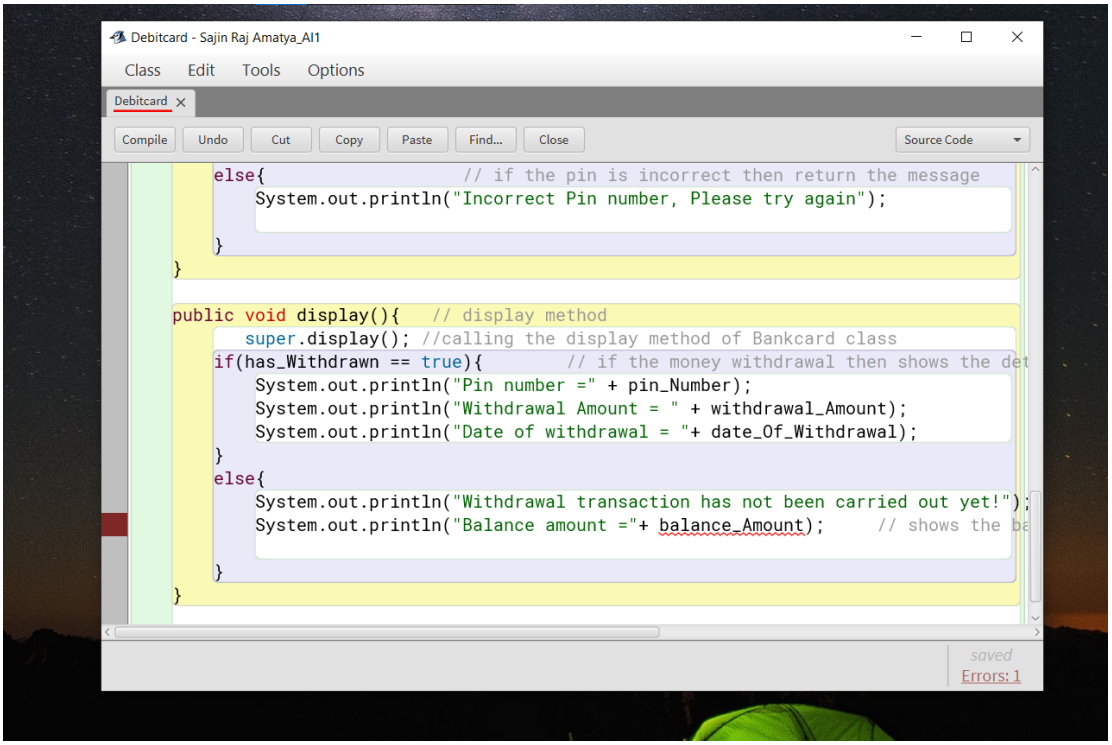


Figure 18 Screenshot of Semantic Error in the program

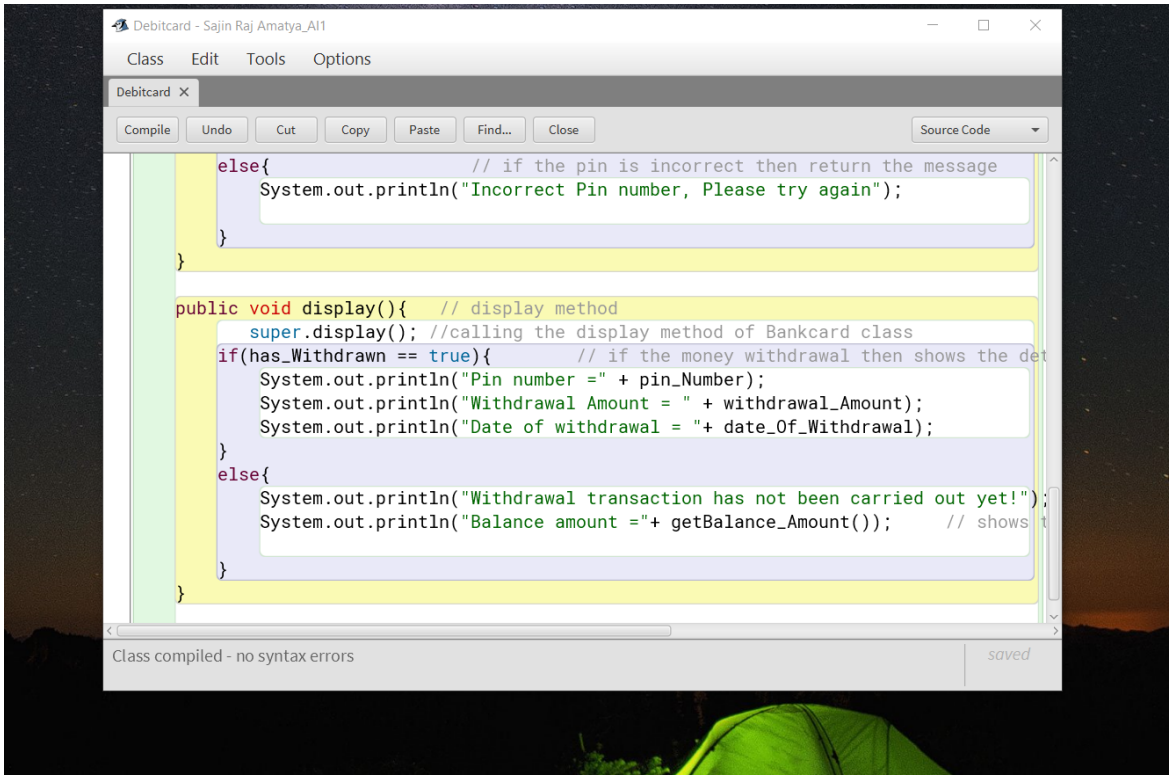


Figure 19 Screenshot of Correction of Semantic Error

7) Conclusion

Therefore, java is a programming language which supports an object-oriented programming paradigm that is based on concept of object which consist of distinct methods and properties, and it can interrelate with one another. With the help of object oriented programming feature such as inheritance and encapsulation that I have learned from the lectures I was able to solve the problem statement of this assignment on time. The concept of OOP was kind of new for me so, I have to get out of my comfort zone and do research on this topic and learn it. I also referred some online source for the idea to complete this assignment. I have also learned the concept of clean code and proper indentation to be followed while writing program and proper documentation. This assignment helped me to get exposure to the practical applicability of an object oriented programing by implementing then it in real word banking system scenario.

While I was doing the assignment I faced a lot of difficulty and challenges in finding the proper solution to the problem statement given to me. I was demotivated and frustrated when I couldn't understand of the proper way of documentation of this assignment. But my module teacher guided me for implementing the proper documentation for this assignment. The concept of OOP encapsulation was challenging for me as I couldn't understand it properly. After I have done a lot of research regarding it, I overcome all those difficulties that I have face during the implementation of OOP feature in this assignment. In Error and detection section, I wasn't able to grasp the concept logical and semantic error that occur in the program. Through the guidance of my module teacher I was able to identify the problem and implement a proper solution to the remove the error in the program.

8) References

Coursera, 2022. *Coursera.* [Online]
Available at: https://www.coursera.org/articles/what-is-java-used-for
[Accessed 20 January 2023].

Cyubahiro, P., 2022. *freeCodeCamp.* [Online]
Available at: https://www.freecodecamp.org/news/object-oriented-programming-concepts-java/
[Accessed 20 January 2023].

Himanshi, 2022. *naukri learning.* [Online]
Available at: https://www.naukri.com/learning/articles/java-methods-explained-with-code/
[Accessed 21 January 2023].

Liang, Y., 2015. *INTRODUCTION TO JAVA PROGRAMMING.* Ninth Edition ed. Boston: PEARSON.

Pepcoding, 2021. *medium.* [Online]
Available at: https://medium.com/@pepcoding/pseudocode-a-brief-article-on-pseudocode-and-its-uses-17965717d733
[Accessed 14 january 2023].

Shankar, R., 2023. *hackr.io.* [Online]
Available at: https://hackr.io/blog/what-is-java#:~:text=Java%20is%20a%20write%2Donce,and%20distributed%20full%2Dfledged%20applications.
[Accessed 20 January 2023].

9) Appendix

9.1) Bankcard Class

```
public class Bankcard   // Parent class name Bankcard is created
{
   private int balance_Amount;        // instance variable with a private access modifier
   private int card_Id;
   private String issuer_Bank;
   private String client_Name;
   private String bank_Account;

   public Bankcard(int balance_Amount, int card_Id, String issuer_Bank, String bank_Account){    //parameterized constructor with 4 parameter
      this.balance_Amount = balance_Amount;  // this keyword is used for invoking the current constructor Bankcard of a class
      this.card_Id = card_Id;
      this.client_Name = "";
      this.issuer_Bank = issuer_Bank;
      this.bank_Account = bank_Account;
   }

   public  int getBalance_Amount(){   //accessor method for balanceamount attribute

      return this.balance_Amount;
   }

   public int  getCard_Id(){   //accessor method for card_Id attribute
```

```
    return this.card_Id;

}

public  String getIssuer_Bank(){   // accessor method for issuer_Bank attribute

    return this.issuer_Bank;
}

public  String getClient_Name(){    // accessor method for client_Name attribute

    return this.client_Name;
}

public  String getBank_Account(){     // accessor method for bank_Account attribute

    return this.bank_Account;

}

public void setBalance_Amount(int balance_Amount){   //setter or mutator  method accessor for balance_Amount attribute
    this.balance_Amount = balance_Amount;

}

public void setClient_Name(String client_Name){        //setter or mutator  method accessor for client_Name attribute
```

```
        this.client_Name=client_Name;
    }

    public void display(){      // Display the enter detail value of a customer
        System.out.println("CardID = "+ this.card_Id);
        if(this.client_Name ==""){          // if condition
            System.out.println("Please enter the client name");
        }
        else{               //else condition
            System.out.println("ClientName = " + this.client_Name);
        }
        System.out.println("IssuerBank = "+ this.issuer_Bank);
        System.out.println("BankAccount = "+ this.bank_Account);
        System.out.println("BalanceAmount = "+ this.balance_Amount);

    }
}
```

9.2) Debitcard child class

```
public class Debitcard extends Bankcard  //subclass Debitcard is created which inherit the property of Bankcard
{
   private int pin_Number;     // instance variable with a private access modifier is declared
   private String date_Of_Withdrawal;
   private int withdrawal_Amount;
   private boolean has_Withdrawn;
                                                    // parameterised constructor of Debitcard
   public Debitcard (int balance_Amount, int card_Id, String bank_Account,String issuer_Bank, String client_Name,int pin_Number){
      super (balance_Amount, card_Id, issuer_Bank, bank_Account); // calling the constructor of superclass Bankcard
      super.setClient_Name(client_Name);// calling the mutator method  of superclass Bankcard
      this.pin_Number = pin_Number;
      this.has_Withdrawn=false;

   }

   public int getPin_Number(){            //accessor method for pin_Number
      return this.pin_Number;
   }

   public int getWithdrawal_Amount(){      //accessor method for  withdrawal_Amount
      return this.withdrawal_Amount;

   }
```

```
    public String getDate_Of_Withdrawal(){ //accessor method for date_Of_Withdrawal
        return this.date_Of_Withdrawal;
    }

    public boolean getHas_Withdrawn(){ // accessor method for has_Withdrawn
        return this.has_Withdrawn;
    }

    public void setWithdrawal_Amount(int withdrawal_Amount){  //mutator method for
withdrawal amount
        this.withdrawal_Amount = withdrawal_Amount;
    }

    public void withdraw(int withdrawal_Amount,String date_Of_Withdrawal,int
pin_Number){  // withdraw method
            this.withdrawal_Amount = withdrawal_Amount;
            this.date_Of_Withdrawal = date_Of_Withdrawal;
        if(this.pin_Number == pin_Number){      // checking if the pin number is correct or
not
            if( withdrawal_Amount<= getBalance_Amount()){  // if correct pin number is
entered then withdrawal amount is checked
                setBalance_Amount(getBalance_Amount()-withdrawal_Amount);   //for the
current balance after the withdrawal
                this.has_Withdrawn = true;
            }
            else{                   // insufficient batlance then return the message
                System.out.println("Not enough balance in your account");
            }
        }
```

```
        else{                   // if the pin is incorrect then return the message
            System.out.println("Incorrect Pin number, Please try again");


        }
    }

    public void display(){   // display method
          super.display(); //calling the display method of Bankcard class
        if(has_Withdrawn == true){        // if the money withdrawal then shows the detail of the transaction
            System.out.println("Pin number =" + pin_Number);
            System.out.println("Withdrawal Amount = " + withdrawal_Amount);
            System.out.println("Date of withdrawal = "+ date_Of_Withdrawal);
        }
        else{
            System.out.println("Withdrawal transaction has not been carried out yet!");
            System.out.println("Balance amount ="+ getBalance_Amount());     // shows the balance amount only


        }
    }

}
```

9.3 Creditcard class

```
public class Creditcard extends Bankcard  // child class credit card which inherit the properties of Bankcard class
{
    private int cvc_Number;                // instance variable with a private access modifier is declared
    private double credit_Limit;
    private double interest_Rate;
    private String expiration_Date;
    private int grace_Period;
    private boolean is_Granted;

    public Creditcard(int card_Id, String client_Name, String issuer_Bank, String bank_Account,    // parameterised constructor of Creditcard
    int balance_Amount, int cvc_Number, double interest_Rate, String expiration_Date){
        super(balance_Amount,card_Id,issuer_Bank,bank_Account);            // calling the constructor of superclass Bankcard
        this.cvc_Number = cvc_Number;
        this.interest_Rate = interest_Rate;
        this.expiration_Date = expiration_Date;
        this.is_Granted = false;
        setBalance_Amount(balance_Amount);            // calling the mutator method  of superclass Bankcard
        setClient_Name(client_Name);
    }
    // getter/accesssor method of each attribute of Creditcard class
    public int getCvc_Number(){
        return this.cvc_Number;
```

```
    }

    public double getCredit_Limit(){
        return this.credit_Limit;
    }

    public double getInterest_Rate(){
        return this.interest_Rate;
    }

    public String getExpiration_Date(){
        return this.expiration_Date;
    }

    public int getGrace_Period(){
        return this.grace_Period;
    }

    public boolean getIs_Granted(){
        return this.is_Granted;
    }
    // method for setting the credit limit
    public void setCredit_Limit(double new_Creditlimit, int new_Graceperiod)
    {
        this.grace_Period = new_Graceperiod;
        if(this.credit_Limit <= 2.5 * getBalance_Amount()){   // if-else condition statment
            is_Granted = true;
            this.credit_Limit = new_Creditlimit;
```

```
        }
        else{
            System.out.println("the credit amount can't be issued,please try again later");

        }
    }

    public void cancel_Credit_Card(){              // method for canceling the creditcard
        this.credit_Limit = 0;
        this.cvc_Number = 0;
        this.grace_Period = 0;
        this.is_Granted = false;
    }
    public void display(){       // method for displaying the detail of Credit card

        super.display();
        System.out.println("CVC Number = " + cvc_Number);
        System.out.println("Interest Rate = "+ interest_Rate);
        System.out.println("Expiration Date = " + expiration_Date);
        if(is_Granted == true){             // if-else statment
            System.out.println("Credit Limit =" + credit_Limit);
            System.out.println("Grace Period =" + grace_Period);
        }
        else{
             System.out.println("Could not display Credit Limit and Grace Period as it is not
permitted");
        }
    }
}
```