CREDIT CARD ADMIN SYSTEM

By

Group two

-Amruth Reddy K

-Anudeep Gorakala

-Deepika Sahu

-Gowtham Reddy Ambavaram

-Sai Sree Vidya Julakanti

-Sonal Shankar Udapudi

-Sivanandalahari Nallapaneni

-Yeswanth Kumar Potnuru

-

CREDIT CARD ADMIN SYSTEM

Abstract

In this project we are mainly focusing on the problem of adding late payment charges to bill amount of their Master Card customers.If the due date is expired , we need to add late payment charges automatically as per given business rules.The proposed system has to look up for Master Card holders and update the Bill Amount by adding late payment charges based on the bill due date as applicable. The Master Card Account details along with updated bill amount have to be persisted in a database.

Introduction

In banking system there are many transactions are done everyday and the employees are need to be observe all the transaction and sometimes it needs to be update at that time it will be very difficult process to look after each and every costumer's status every day soo to reduce that complexity in this work we are introducing automatic process to update the adding late payment charges to bill amount of their Master Card Customers when the due date is expired. In this work we need to look up for Master Card holders and update the bill amount by adding late payment charges based on the business rules so that we can able to reduce the man power and also we can able to make the customer's status uptodate.

Technologies used (or) Requirements

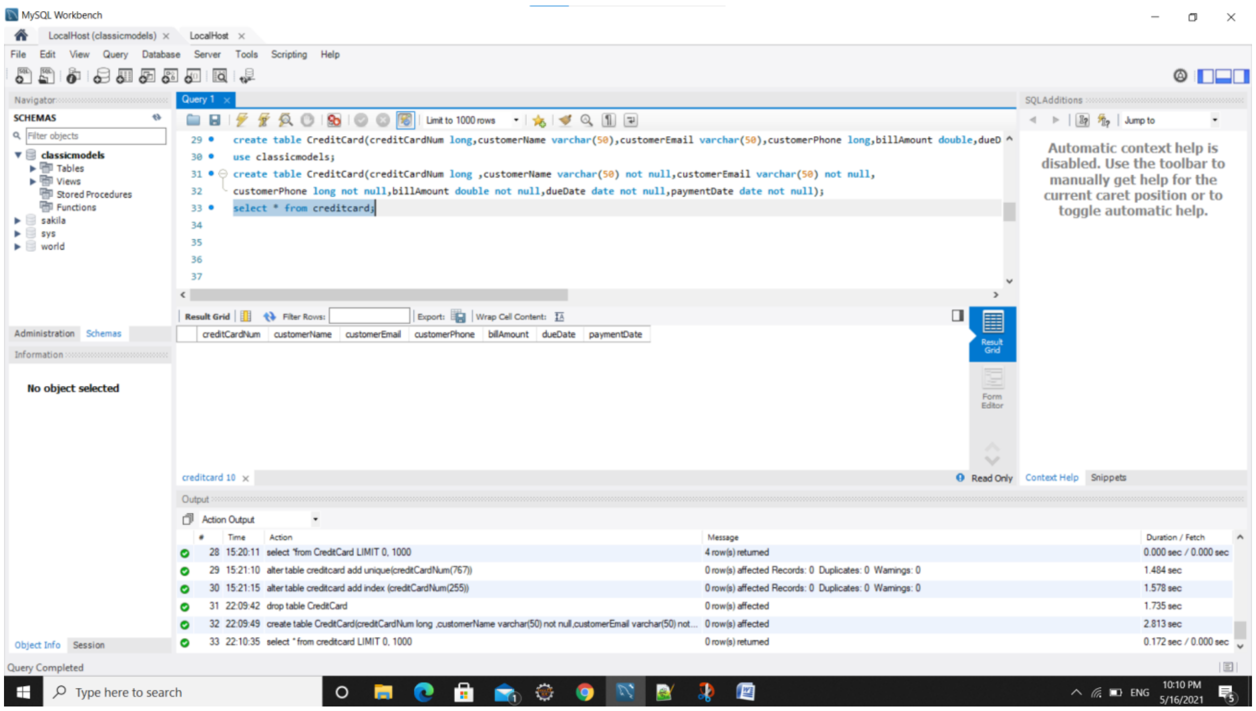
->MYSQL

->JAVA

->MAVEN

->JUnit

MYSQL: In order to store the customer's details we are using mysql database .Here in this project we are creating a table named CreditCard and it contains cloums i.e., CreditCardNum , CustomerName , CustomerEmail , CustomerPhone , BillAmount , DueDate , PaymentDate.Empty table without any values is as follows:



```
create table CreditCard(creditCardNum long,customerName varchar(50),customerEmail varchar(50),customerPhone long,billAmount double,dueD
use classicmodels;
create table CreditCard(creditCardNum long ,customerName varchar(50) not null,customerEmail varchar(50) not null,
customerPhone long not null,billAmount double not null,dueDate date not null,paymentDate date not null);
select * from creditcard;
```

creditCardNum	customerName	customerEmail	customerPhone	billAmount	dueDate	paymentDate

```
#   Time      Action                                                                                    Message                                                  Duration / Fetch
28  15:20:11  select *from CreditCard LIMIT 0, 1000                                                     4 row(s) returned                                        0.000 sec / 0.000 sec
29  15:21:10  alter table creditcard add unique(creditCardNum(767))                                     0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.484 sec
30  15:21:15  alter table creditcard add index (creditCardNum(255))                                     0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.578 sec
31  22:09:42  drop table CreditCard                                                                     0 row(s) affected                                        1.735 sec
32  22:09:49  create table CreditCard(creditCardNum long ,customerName varchar(50) not null,customerEmail varchar(50) not...  0 row(s) affected     2.813 sec
33  22:10:35  select * from creditcard LIMIT 0, 1000                                                    0 row(s) returned                                        0.172 sec / 0.000 sec
```

JAVA

maven: It is a build automation tool used primarily for java projects. Maven can also be used to build and manage projects.

In order to PARSEINPUT we need the below files

->CreditCardAdminService.java,

-> ApplicationUtil.java,

-> CreditCard.java,

->Inputfeed.txt,

->CreditCardAdminSystemException.java

These files are used to parse Input file, and convert into value objects.

In order to check whether it is the mastercard or not we reads the input file, does validation to check if the record is Master Card, builds the Credit Card value object and returns it .Here we are following the below constraints to validate that

a) Input file format is .txt and is comma separated (Sample rows are added. You can add any number of rows to test your service class, from main method.

c) File Structure is like below:

<credit card number>,<customer name>,<customer email>,<customer phone>,<bill amount>,<due date>,<paid date>

d) In the input feed, filter customers who hold only Master card. You can identify the master card holder by checking if credit card number of the row starts with '5'.

e) Assume that the bill amount is in INR

f) Assume that the Due Date or Paid Date in the file will be in the format yyyy-MM-dd.

g) Do not change the data types of the value object given in POJO.

h) Always convert the due date and paid date values to java.util.date with format, yyyy-MM-dd before setting in CreditCard value object.

i) Use ApplicationUtil.java for reading file, performing date operations, etc.

Process Flow:

a) The app will be invoked by calling the CreditCardAdminService. addCreditCardDetails with the input feed (.txt file)

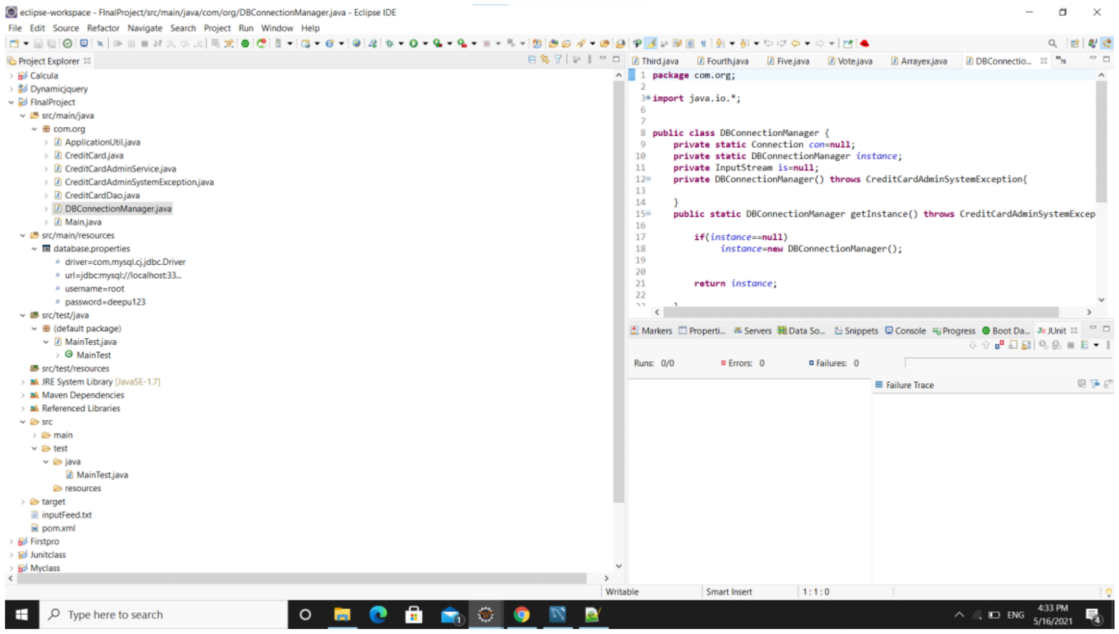
b) Read the file using File I/O or Java Streams in ApplicationUtil. readFile method

c) Return a list of Master Card rows from input file, from the readFile method Credit Card Admin System

d) Code the method CreditCardAdminService.buildMasterCreditCardList. Call the readFile method from this method. Read every line from the list returned by readFile method, split the records based on comma separator

e) Use the ApplicationUtil. convertStringToDate method to convert the date from String Format to java.util.Date format (yyyy-MM-dd).

f) Build the Credit Card Value Object from the values obtained in every line .


eclipse-workspace - FInalProject/src/main/java/com/org/DBConnectionManager.java - Eclipse IDE

```
Project Explorer
> Calcula
> Dynamicjquery
v FInalProject
  v src/main/java
    v com.org
      > ApplicationUtil.java
      > CreditCard.java
      > CreditCardAdminService.java
      > CreditCardAdminSystemException.java
      > CreditCardDao.java
      > DBConnectionManager.java
      > Main.java
  v src/main/resources
    v database.properties
        driver=com.mysql.cj.jdbc.Driver
        url=jdbc:mysql://localhost:33...
        username=root
        password=deepu123
  v src/test/java
    v (default package)
      v MainTest.java
        > MainTest
    src/test/resources
  > JRE System Library [JavaSE-1.7]
  > Maven Dependencies
  > Referenced Libraries
  v src
    > main
    v test
      v java
          MainTest.java
        resources
  > target
    inputFeed.txt
    pom.xml
> Firstpro
> Junitclass
> Myclass
```

```java
package com.org;

import java.io.*;


public class DBConnectionManager {
    private static Connection con=null;
    private static DBConnectionManager instance;
    private InputStream is=null;
    private DBConnectionManager() throws CreditCardAdminSystemException{

    }
    public static DBConnectionManager getInstance() throws CreditCardAdminSystemExcep

        if(instance==null)
            instance=new DBConnectionManager();


        return instance;

```

```
Runs: 0/0    Errors: 0    Failures: 0
```

Persist Data into Database

Here we are calculate the updated bill amount and add card details to database it follows some constraints

a) The database.properties has connection details required to connect to the backend

b) Do not change the keys of the property files, you can update the values based on the local database settings. For example, do not change the key, db.username. Rather you can have any value as user name based on local settings.

c) Use only JDBC to establish Database connection

d) Assume the location of the property file will be always as given in the skeleton.

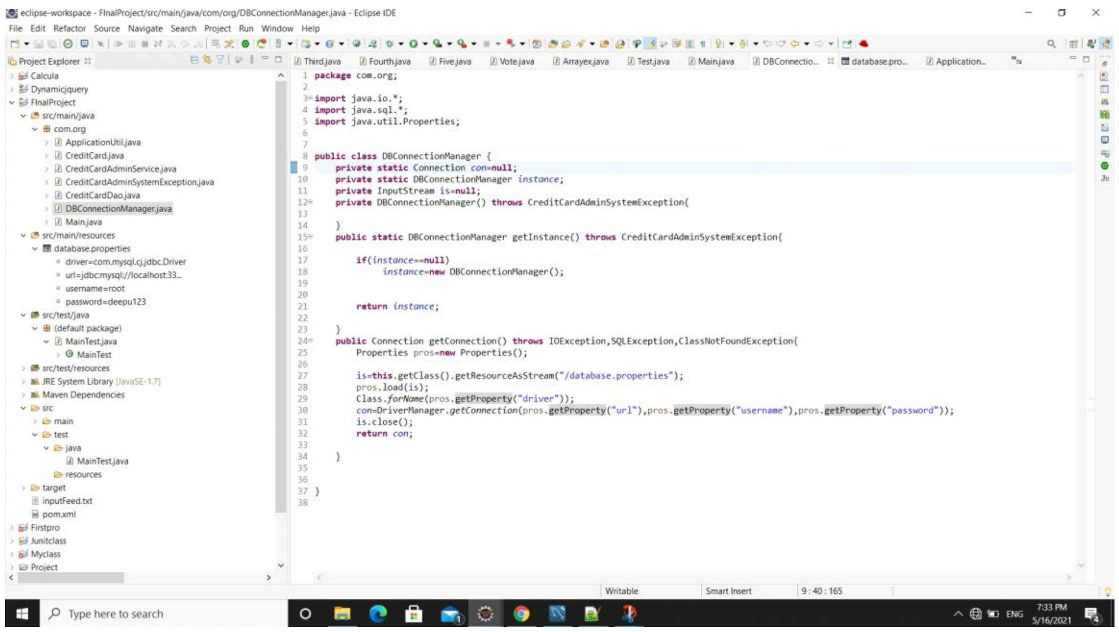
e) Don't Hardcode the connection string to establish database connection. Read it from property files.

f) Use Prepared Statement to insert records

g) Close all the resources after use

h) Catch all database related exception and throw Application specific exception only from DAO or from DBConnectionManager class. There has to be a private constructor in DBConnectionManager class, to load the database property file and to establish a database connection using JDBC

i) Rollback the Insert if any SQL exception has occurred. Throw application specific exception.


eclipse-workspace - FinalProject/src/main/java/com/org/DBConnectionManager.java - Eclipse IDE
File Edit Refactor Source Navigate Search Project Run Window Help
Project Explorer
Third.java Fourth.java Five.java Vote.java Arrayex.java Test.java Main.java DBConnectio... database.pro... Application...
Writable
Smart Insert
9 : 40 : 165
Type here to search

```
Calcula
Dynamicjquery
FinalProject
  src/main/java
    com.org
      ApplicationUtil.java
      CreditCard.java
      CreditCardAdminService.java
      CreditCardAdminSystemException.java
      CreditCardDao.java
      DBConnectionManager.java
      Main.java
  src/main/resources
    database.properties
      driver=com.mysql.cj.jdbc.Driver
      url=jdbc:mysql://localhost:33...
      username=root
      password=deepu123
  src/test/java
    (default package)
      MainTest.java
        MainTest
  src/test/resources
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  src
    main
    test
      java
        MainTest.java
      resources
  target
  inputFeed.txt
  pom.xml
Firstpro
Junitclass
Myclass
Project
```

```java
package com.org;

import java.io.*;
import java.sql.*;
import java.util.Properties;


public class DBConnectionManager {
    private static Connection con=null;
    private static DBConnectionManager instance;
    private InputStream is=null;
    private DBConnectionManager() throws CreditCardAdminSystemException{

    }
    public static DBConnectionManager getInstance() throws CreditCardAdminSystemException{

        if(instance==null)
            instance=new DBConnectionManager();


        return instance;

    }
    public Connection getConnection() throws IOException,SQLException,ClassNotFoundException{
        Properties pros=new Properties();

        is=this.getClass().getResourceAsStream("/database.properties");
        pros.load(is);
        Class.forName(pros.getProperty("driver"));
        con=DriverManager.getConnection(pros.getProperty("url"),pros.getProperty("username"),pros.getProperty("password"));
        is.close();
        return con;

    }


}
```

Process Flow

a) Modify the CreditCardAdminService.buildMasterCreditCardList method check if the due date is expired (past date). If yes, add 500 INR to the bill amount, and set the updated bill amount to the Master Credit Card objects. Credit Card Admin System

b) Use CreditCardAdminService .getBillAmountWithLatePaymentCharges method to add 500 to the bill amount passed as parameter, based on due date.

c) The method CreditCardAdminService .buildMasterCreditCardList must return the list of master cards with updated bill amount, after adding late payment charges as applicable. For records where the due date is not expired, let’s persist the records as it is.

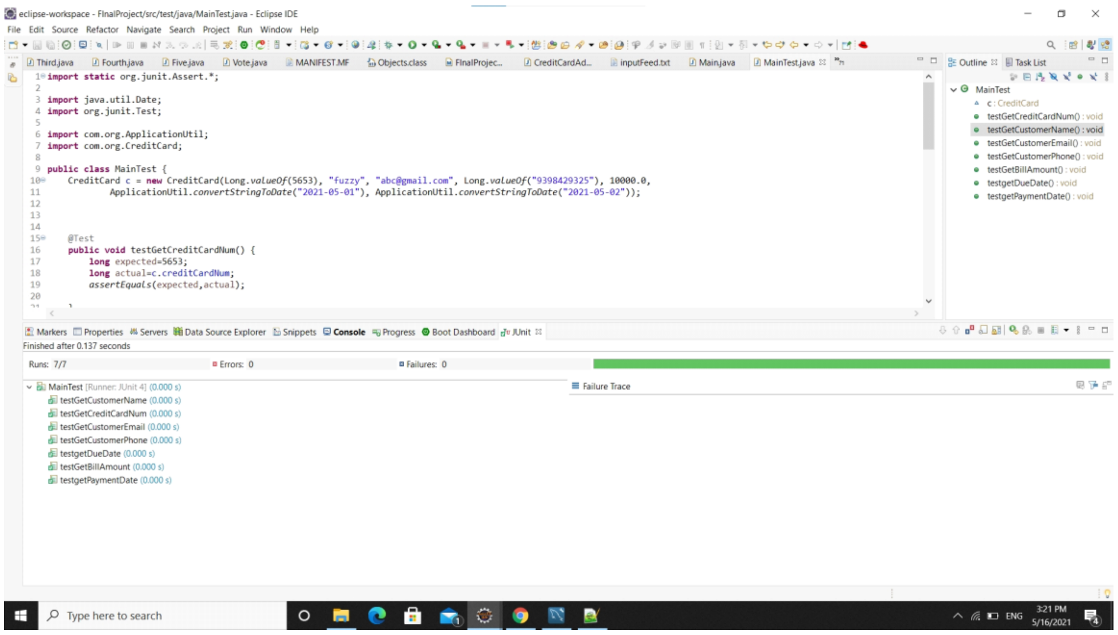
d) After reading file, building records as List<CreditCard>, call the CreditCardDAO. addCreditCardDetails method to insert values to database. You may have to convert the java.util.date to java.sql.date before storing to database.

e) If Insert has happened successfully, return true; false otherwise

JUnit

Junit is an open source framework, which is used for writing and running tests. It provides annotations to identify test methods, assertions for testing expected results and provides test runners for running tests.

Here we are creating junit file named as MainTest.java to test our actual results with the expected results. It can be shown as follows



eclipse-workspace - FinalProject/src/test/java/MainTest.java - Eclipse IDE

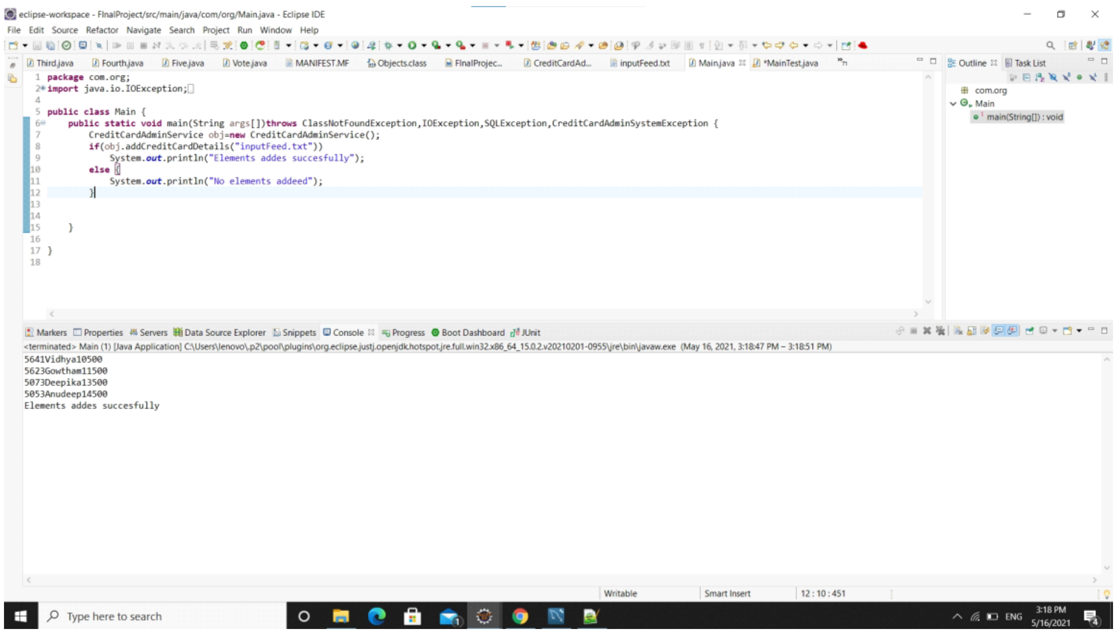
```
import static org.junit.Assert.*;

import java.util.Date;
import org.junit.Test;

import com.org.ApplicationUtil;
import com.org.CreditCard;

public class MainTest {
    CreditCard c = new CreditCard(Long.valueOf(5653), "fuzzy", "abc@gmail.com", Long.valueOf("9398429325"), 10000.0,
            ApplicationUtil.convertStringToDate("2021-05-01"), ApplicationUtil.convertStringToDate("2021-05-02"));



    @Test
    public void testGetCreditCardNum() {
        long expected=5653;
        long actual=c.creditCardNum;
        assertEquals(expected,actual);

```

```
Finished after 0.137 seconds
Runs: 7/7    Errors: 0    Failures: 0

MainTest [Runner: JUnit 4] (0.000 s)
    testGetCustomerName (0.000 s)
    testGetCreditCardNum (0.000 s)
    testGetCustomerEmail (0.000 s)
    testGetCustomerPhone (0.000 s)
    testgetDueDate (0.000 s)
    testGetBillAmount (0.000 s)
    testgetPaymentDate (0.000 s)
```

we are checking all the test cases to verify whether it satisfies our requirements or not.

Final output:

Once all the work is done the final output can be displayed by executing the file named Main.java then the output can be displayed as follows

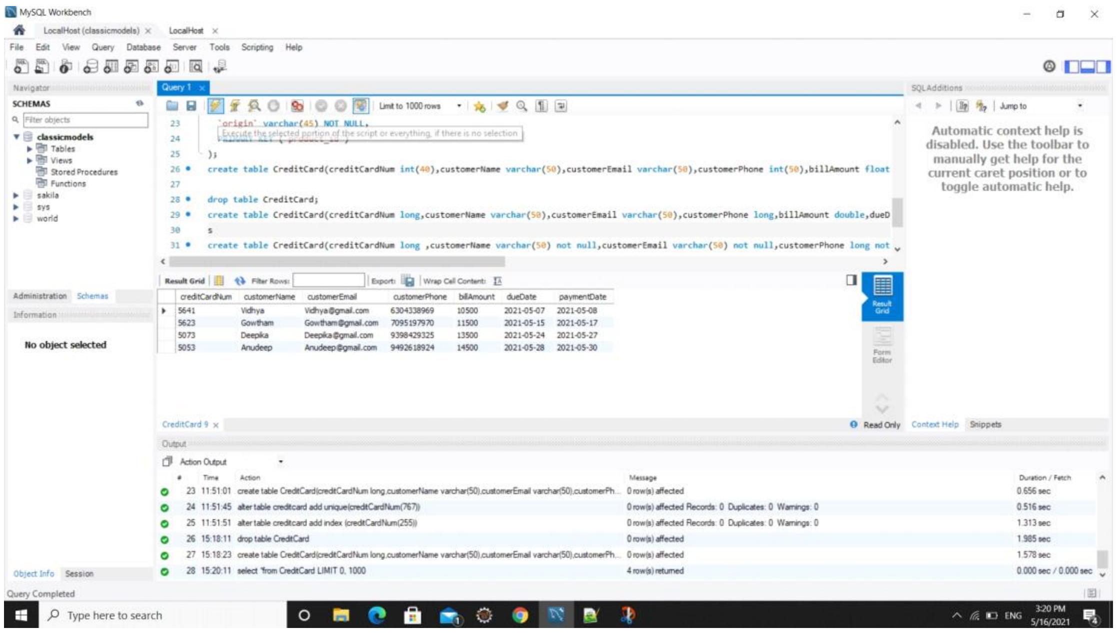


eclipse-workspace - FinalProject/src/main/java/com/org/Main.java - Eclipse IDE

```java
package com.org;
import java.io.IOException;

public class Main {
    public static void main(String args[])throws ClassNotFoundException,IOException,SQLException,CreditCardAdminSystemException {
        CreditCardAdminService obj=new CreditCardAdminService();
        if(obj.addCreditCardDetails("inputFeed.txt"))
            System.out.println("Elements addes succesfully");
        else {
            System.out.println("No elements addeed");
        }


    }

}
```

```
<terminated> Main (1) [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe  (May 16, 2021, 3:18:47 PM – 3:18:51 PM)
5641Vidhya10500
5623Gowtham11500
5073Deepika13500
5053Anudeep14500
Elements addes succesfully
```

Finally the table in database has been updation with total bill amount to be paid by each master card holder when card holder missed the deadline was successful .Updations on table CreditCared is as follows:



```
  `origin` varchar(45) NOT NULL,
  Execute the selected portion of the script or everything, if there is no selection
);
create table CreditCard(creditCardNum int(40),customerName varchar(50),customerEmail varchar(50),customerPhone int(50),billAmount float

drop table CreditCard;
create table CreditCard(creditCardNum long,customerName varchar(50),customerEmail varchar(50),customerPhone long,billAmount double,dueD
s
create table CreditCard(creditCardNum long ,customerName varchar(50) not null,customerEmail varchar(50) not null,customerPhone long not
```

creditCardNum	customerName	customerEmail	customerPhone	billAmount	dueDate	paymentDate
5641	Vidhya	Vidhya@gmail.com	6304338969	10500	2021-05-07	2021-05-08
5623	Gowtham	Gowtham@gmail.com	7095197970	11500	2021-05-15	2021-05-17
5073	Deepika	Deepika@gmail.com	9398429325	13500	2021-05-24	2021-05-27
5053	Anudeep	Anudeep@gmail.com	9492618924	14500	2021-05-28	2021-05-30

```
#   Time      Action                                                                                          Message                                                    Duration / Fetch
23  11:51:01  create table CreditCard(creditCardNum long,customerName varchar(50),customerEmail varchar(50),customerPh...  0 row(s) affected                                          0.656 sec
24  11:51:45  alter table creditcard add unique(creditCardNum(767))                                            0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.516 sec
25  11:51:51  alter table creditcard add index (creditCardNum(255))                                           0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   1.313 sec
26  15:18:11  drop table CreditCard                                                                           0 row(s) affected                                          1.985 sec
27  15:18:23  create table CreditCard(creditCardNum long,customerName varchar(50),customerEmail varchar(50),customerPh...  0 row(s) affected                                          1.578 sec
28  15:20:11  select *from CreditCard LIMIT 0, 1000                                                           4 row(s) returned                                          0.000 sec / 0.000 sec
```

Division of work:

Java coding part:

- Deepika Sahu

-Sai Sree Vidya Julakanti

-Gowtham Reddy Ambavaram

-Amruth Reddy K

Testing and Documentation:

-Anudeep Gorakala

-Yeswanth Kumar Potnuru

Database and Documentation:

-Sivanandalahari Nallapaneni

-Sonal Shankar Udapudi