Lambton College

Class:	MAD_3463_3_
HARMANJEET SINGH	790083
SUKHRAJ KAUR	784511
REKHA RANI	785502
SANDEEP KAUR	785503

BANK MANAGEMENT PROJECT

○ Index

○ Introduction a brief about the system

This group has created a bank management system in which the user is able to create multiple accounts and of more than one type . The user has the access to change the name in the account details . The various transactions like debiting the account, depositing money in the account , transfer the data from the account to an another account and ay utilities bills are done by the user in this system . The java program create a text file in which all the specifications of an account has been added and saved into the storage unit. The specifications includes the name , account no. , account type selected by the user, the yearly charges , and the balance in the account .

○ Full description of the system including the minimum requirements you have up at the above description

System is generated by the java language using eclipse platform. In the system we have to create three classes and many methods along with the main method.

The Array list is used to hold the various objects of the classes where account is created so that the system can hold multiple accounts and the number of accounts should not be limited.

Various methods are created to perform many transactions .

The constructors are made to give the values to the classes .

To create the text files ,to read the files and to change the data in the text file File class is imported.

Joption Class is imported to get the pop up boxes and input boxes so that the user interface would be friendly and easy.

Input output Throws are applied to overcome the exemptions.

Conditional statements are used to made decisions at various stages.

Loops are used to perform multiple tasks in one stretch .

○ Explain the classes with their attributes and methods (not code)

In the BankAccount class there are generally the methods to do the transactions in the account . Balance attribute is there of double type which is set by the starting balance variable with the constructor and overloading is used to get different types of data. Methods in this class have all variables, and parameters of double type and get the input at the time when they are called in the main method.

Deposit – Used to add amount to the balance.

Withdraw- To withdraw the amount of money from the account . In this method the balance is reduced by the amount withdrawn. And amount of 2 is also deducted as transaction charges.

Transfer- To transfer the money to an another account the system has transfer method The method reduced the amount of money from the balance which is transferred.

PayutilityBills – the system has payUtilityBills method to pay the utility bills. This method reduce the amount of money from the balance.

Then the class has setbalance method that takes a parameter and assigns it to variable balance and getbalance methods which is used to return the values of the variable balance

In the Customer class the system has constructor that sets the values of the parameters which are passed by the new objects . The variable are bankaccount, name , balance , interest , charges

Then the method named showData is there that print the values of all the details of an account.

The filecreate method created a new file in the system and write all the specifications of an account in the text file .

The **one.java class** has the main method in which all the functions of the systems are there, all the methods are called here in this main method., All the inputs from the user is taken in this method.

Alongwith this there are some methods created in this class which are used .like

accounType method which when called get the input from the user for the type of account he wants to select through the Joption box input and return the value.

customerName method take the input from the customer through pop up window input panel and return the name of the customer when it is called .

Then the system has a method named b**alanceMethod** in which an object is created for the class BankAccount and all the methods of the BankAccount class are clled here in this method. So it is kind of parent method which called different other methods. It return the balance in the last to the main method where it is called .

There is **modifyFile** method which is used to change the name of the customer in an account. In this method the BufferedReader is used to alter the text in the file . It takes the old name from the file and switch it with the new name.

○ Explain exactly what the system can do (all tasks)

The system **Bank Management** is created to perform a plenty of tasks which are done in series . In the system the user can create a new bank account of any type which is available under which he choose his name , the starting balance .

User can create multiple no. of accounts .

The user can make the transactions in the account like withdraw, deposit, transfer to another account and pay utility bills.

Account details are saved in the Text file in .txt format under the name of the account no.

The user can see the account details from the .txt file having his account details

The user has an option to change the account details and save it again in the .txt file.

○ User manual

Run the one.class file having main method

1.To create an account

Step 1. When the system ask “whether you have an account in the bank or not”
Type **false** in the pop up box.

Step 2. Type 1 for selecting the **saving** account type **1**
(Saving account give interest rate of 5% and yearly charges & 100)
Or Type **2** for checkin account

(Check in account give interest rate of 2% and yearly charges & 200)

Step 3. Enter the name of the account holder
Step 4. Enter the starting balance of an account
Step 5. Enter the amount to be withdrawn
Step 6. Enter the amount to be deposited
Step 7. Enter the amount to be transferred to an another account
Step 8. Enter the amount want to pay for utility bills.

a random account no is generated and .txt file is saved under the name of account no. having the information related account details.

2.To view an account details

Step 1. When the system ask “whether you have an account in the bank or not”
Type **true** in the pop up box.
Step 2. Type the **account no.**
Step 3. Type **false** in the pop up box to see the details .
Account details will be shown in the compiler

3.To edit an account details

Step 1. When the system ask “whether you have an account in the bank or not”
Type **true** in the pop up box.
Step 2. Type the **account no.**
Step 3. Type **true** in the pop up box to see the details and edit the name
Step 4. Type the previous name you want to edit.
Step 5. Type the new name .
Name of the account holder will be changed and shown in the compiler and also the .txt file has been edited

○ The Source Code

Class having main method ----

```
package bankmanagment;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

import javax.swing.JOptionPane;

public class one {


    /**
     * @param args
     * @throws FileNotFoundException
     */
    public static void main(String[] args) throws FileNotFoundException {
            //create array of customer object

      ArrayList<Customer> obj = new ArrayList<Customer>();
      ////////////////
            JOptionPane.showMessageDialog(null, "Welcome to the Bank Account
System");
            String continu=null;


            boolean bankAccountAsk; //Boolean for asking if user have  bank account
```

```
                String inputbankAccountAsk; // string to hold the input
                inputbankAccountAsk = JOptionPane.showInputDialog("Enter true if you
have an account in bank or type false");

                bankAccountAsk =Boolean.parseBoolean(inputbankAccountAsk);
                // variables
                int id1;
                double balance = 0;
                int interestRate ;
                int charges;
                String name1,InputDialogue,InputDialogue2;
                // condition loop to see details and edit already existing account
                if (bankAccountAsk) {
                        JOptionPane.showMessageDialog(null, "Welcome to the  Bank");
                        String filename;

                        filename = JOptionPane.showInputDialog("enter account no to
see the account details ");

                        //// open the file with account no
                        File firstFile = new File(filename+".txt");///showing text
                                Scanner firstFileRead = new Scanner(firstFile);
                                while(firstFileRead.hasNext())
                                {
                                        String data = firstFileRead.nextLine();
                                        System.out.println(data);
                                }

                                //// edit data

                                // variables for editing the data in .txt file
                                String editData;
                                Boolean EditData;
                                JOptionPane.showMessageDialog(null, "If you want to edit
the data press true else type false");
                                editData = JOptionPane.showInputDialog("If you want to
edit the data press true else type false");
                        EditData= Boolean.parseBoolean(editData);
                  if (EditData) {
                       String fil=filename+".txt";
```

```
                              String oldName = JOptionPane.showInputDialog("Type previous
name ");// previous name input

                              String newName = JOptionPane.showInputDialog("Type new
name ");// new name input


                               modifyFile(fil, oldName, newName);/// method calling for modify

                               System.out.println("Modified Account Details are following");


                               // showing the edited file

                               File firstFile1 = new File(filename+".txt");///showing text
                                          Scanner firstFileRead1 = new Scanner(firstFile1);
                                          while(firstFileRead1.hasNext())
                                          {
                                                     String data = firstFileRead1.nextLine();
                                                     System.out.println(data);
                                          }


                   }
                   else {
                              JOptionPane.showMessageDialog(null, "Thanks ");
                   }
                    }
                    else/// To create a new account
                    {
                               JOptionPane.showMessageDialog(null, "Create a new Bank
Account");
                               Random random = new Random();
                               int accountNO = random.nextInt(800000);


                                          id1=accountType(); // calling method to get the account
type
                                          if  (id1==1) {
                                                     interestRate= 5;  // setting interset rate and chrges
for saving acc
```

```
                                    charges=100;
                                }
                                else
                                { interestRate=2;    // setting interset rate and chrges for
checkin acc
                                charges = 200;}


                                name1=customerName(); // getting customer name by
calling method

                                balance= balanceMethod();    //
                                 obj.add(new
Customer(accountNO,name1,balance,interestRate,charges));



                }
                        // making account as object and assingning it values of obj
arraylist element


                          for (int index = 0; index < obj.size(); index++){
                                 Customer account = obj.get(index);
                                 account.showData();// showdata method called from
employee
                                 account.fileCreate();// file create method call from
employee
                                 }
                          //String continu1;
                                //continu1 = JOptionPane.showInputDialog("type exit to
exit ");


                                //continu=continu1;


                           // } while (continu!= "exit");
                        }

    // method for account type selection
```

```
        public static int accountType()
         {
                int id1;
                String input;
                input = JOptionPane.showInputDialog("Please enter 1 for saving account
2 for checkin account");

                id1= Integer.parseInt(input);

                return id1;

         }

        // method for customer name input
        public static String customerName()
         {
                String name1;
                String input;
                input = JOptionPane.showInputDialog("Please enter Account Holder
Name then press enter");

                name1= input;


                return name1;
         }
        /// method to set the balance and perform all transactions
        public static double balanceMethod()
        {
            String input;    // To hold user input
double balance;
            // Create a DecimalFormat object for displaying dollars.
            DecimalFormat dollar = new DecimalFormat("#,###.00");

            // Get the starting balance.
            input = JOptionPane.showInputDialog("What is your " +
                            "account's starting balance?");

            // Create a BankAccount object.
            BankAccount account = new BankAccount(input);
            // Get the amount of pay.
            input = JOptionPane.showInputDialog("How much you " +
                              "you want to deposit ");

            // Deposit the user's deposit into the account.
            account.deposit(input);

            // Display the new balance.
            JOptionPane.showMessageDialog(null,
                         "Your amount has been  deposited.\n" +
                         "Your current balance is $ " +
                         dollar.format(account.getBalance()));

            // Withdraw some cash from the account.
            input = JOptionPane.showInputDialog("How much would " +
                              "you like to withdraw? ");


            // if loop to check the balance greater than withdraw aount
            if (Double.parseDouble(input)<=account.getBalance()+2)
            {
                account.withdraw(input);

            // Display the new balance
            JOptionPane.showMessageDialog(null,
                       "Now your balance after chrges of $2 is $" +
                       dollar.format(account.getBalance()));
            }
            else  JOptionPane.showMessageDialog(null,"Balance  is  less  than  withdraw
amount");
      // transfer some cash from the account.
            input = JOptionPane.showInputDialog("How much would " +
                              "you like to transfer to another account ");



            if  (Double.parseDouble(input)<=account.getBalance())//  to  check  balance
greater than transfer amount
            {

            account.transfer(input);
```

```
            // Display the new balance
            JOptionPane.showMessageDialog(null,
                        "Now your balance after transfer is $" +
                        dollar.format(account.getBalance()));
            }
            else  JOptionPane.showMessageDialog(null,"Balance  is  less  than  transfer
amount");


            // pay utility bills from the account.
            input = JOptionPane.showInputDialog("How much would " +
                                    "you like to pay for utility bills? ");
            if (Double.parseDouble(input)<=account.getBalance())// to check the balance
greater than utility bills
            {


            account.utilityBills(input);

            // Display the new balance
            JOptionPane.showMessageDialog(null,
                        "Now your balance after Bills is $" +
                        dollar.format(account.getBalance()));
            }
            else
                  JOptionPane.showMessageDialog(null,"Balance is less than utility bills
amount");


balance=account.getBalance();

                 return balance;

       }
       /// method for modify the account details in the  .txt file
       static void modifyFile(String filePath, String oldString, String newString)
    {
       File fileToBeModified = new File(filePath);
```

```
String oldContent = "";

BufferedReader reader = null;

FileWriter writer = null;

try
{
    reader = new BufferedReader(new FileReader(fileToBeModified));

    //Reading all the lines of input text file into oldContent

    String line = reader.readLine();

    while (line != null)
    {
        oldContent = oldContent + line + System.lineSeparator();

        line = reader.readLine();
    }

    //Replacing oldString with newString in the oldContent

    String newContent = oldContent.replaceAll(oldString, newString);

    //Rewriting the input text file with newContent

    writer = new FileWriter(fileToBeModified);

    writer.write(newContent);
}
catch (IOException e)
{
    e.printStackTrace();
}
finally
{
    try
    {
        //Closing the resources
```

```
                reader.close();

                writer.close();
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
        }
    }






      }
```

Second Class

```
package bankmanagment;

import java.io.FileNotFoundException;
import java.io.PrintWriter;

public class Customer {

private double balance;      // new Account balance

            int bankAccount;
            String name;
            int interest;
            int charges;
```

```
  //Employee class constructor
  public Customer(int bAcc, String customerName, double balance2, int intr, int chrg){
     bankAccount = bAcc;
     name = customerName;
     balance=balance2;
     interest = intr;
     charges=chrg;

  }

  // method to show the account details
 public void showData(){
    System.out.print("ACCOUNT NO is= "+bankAccount  + "\n  " + " CUSTOMER Name = "+name +"\n"+
                          "Balance is  " +balance+" \n"+" interest rate is  "+interest+"\n"+
                          "Yearly charges are "+ charges +"\n"+"Withdrawl charges are $2 everytime")
    ;
    System.out.println();
  }



 // method to create a .txt file

 public void fileCreate() throws FileNotFoundException {

             PrintWriter file2 = new PrintWriter(bankAccount+".txt"  );
             file2.println("Account no is"+bankAccount);
             file2.println("Name of customer :"+name);
             file2.println("BAlance of an account:"+balance);
             file2.println("interest rate for account :"+interest);
             file2.println("Yearly charges for the account "+charges);
             file2.println("withdrawl charges  $2 everytime ");

             file2.close();



  }

}
```

Third class

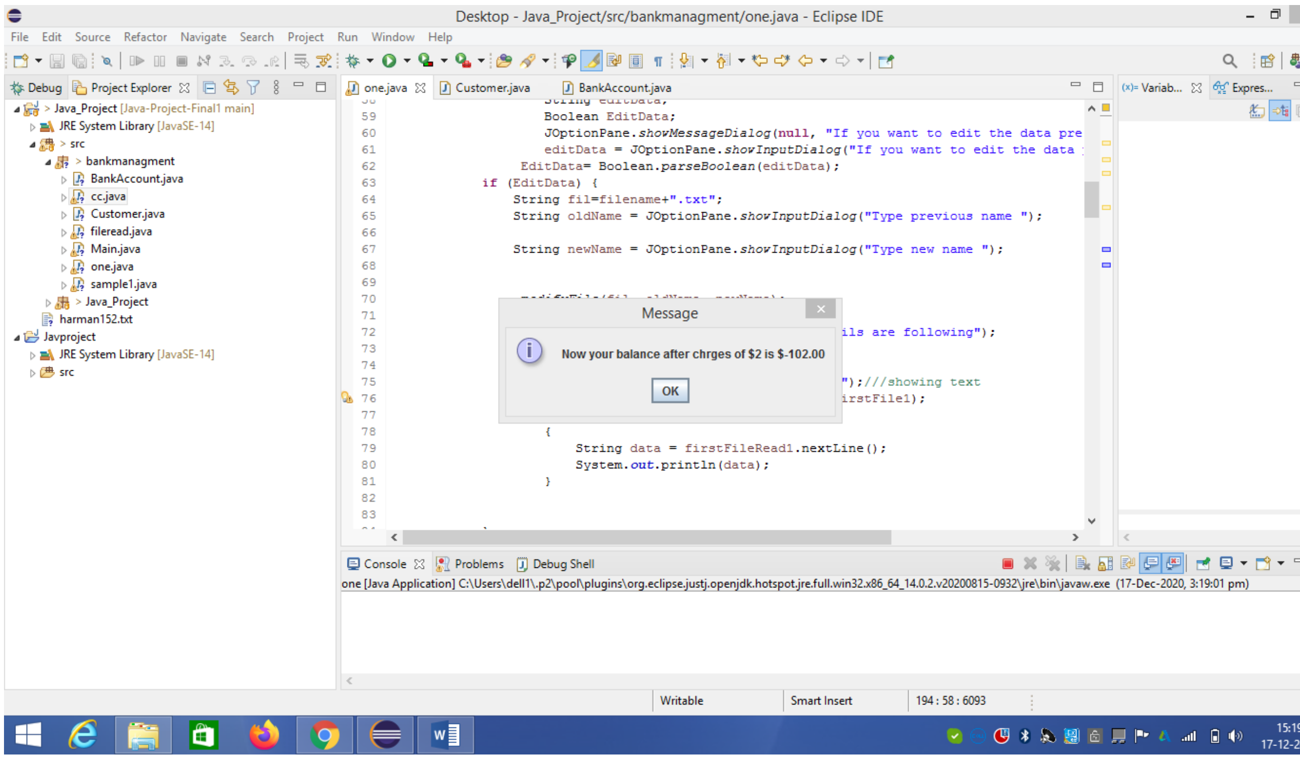
```
package bankmanagment;
/**
   The BankAccount class simulates a bank account.
*/

public class BankAccount
{
   private double balance;      // Account balance

   /**
      This constructor sets the starting balance
      at 0.0.
   */

   public BankAccount()
   {
      balance = 0.0;
   }

   /**
      This constructor sets the starting balance
      to the value passed as an argument.
      @param startBalance The starting balance.
   */

   public BankAccount(double startBalance)
   {
      balance = startBalance;
   }

   /**
      This constructor sets the starting balance
      to the value in the String argument.
      @param str The starting balance, as a String.
   */

   public BankAccount(String str)
   {
      balance = Double.parseDouble(str);
   }

   /**
      The deposit method makes a deposit into
      the account.
      @param amount The amount to add to the
                    balance field.
   */

   public void deposit(double amount)
   {
      balance += amount;
```

```
}

/**
  The deposit method makes a deposit into
  the account.
  @param str The amount to add to the
           balance field, as a String.
*/

public void deposit(String str)
{
  balance += Double.parseDouble(str);
}

/**
  The withdraw method withdraws an amount
  from the account.
  @param amount The amount to subtract from
             the balance field.
*/

public void withdraw(double amount)
{

           balance -= amount;
  balance-=2;
}

/**
  The withdraw method withdraws an amount
  from the account.
  @param str The amount to subtract from
           the balance field, as a String.
*/

public void withdraw(String str)
{
  balance -= Double.parseDouble(str);
  balance-=2;
}
public void transfer(double amount)
{
  balance -= amount;

}

/**
  The transfer method pay utility bills
  @param str The amount to subtract from
           the balance field, as a String.
*/
public void utilityBills(String str)
{
  balance -= Double.parseDouble(str);
```

```
   }
   public void utilityBills(double amount)
   {
     balance -= amount;

   }

   /**
     The transfer method transfer an amount
     from the account.
     @param str The amount to subtract from
            the balance field, as a String.
   */
   public void transfer(String str)
   {
     balance -= Double.parseDouble(str);
   }

   /**
     The setBalance method sets the account balance.
     @param b The value to store in the balance field.
   */

   public void setBalance(double b)
   {
     balance = b;
   }

   /**
     The setBalance method sets the account balance.
     @param str The value, as a String, to store in
            the balance field.
   */

   public void setBalance(String str)
   {
     balance = Double.parseDouble(str);
   }

   /**
     The getBalance method returns the
     account balance.
     @return The value in the balance field.
   */

   public double getBalance()
   {
     return balance;
   }
 }
```

○ Some screenshots showing the errors or exceptions you get while you are testing the system, and explain how you could solve them.

While withdraw , transfer , and pay utility bills the balance show –ve if amount is filled more than balance which is not acceptable in the banking system .



Desktop - Java_Project/src/bankmanagment/one.java - Eclipse IDE

```
                Boolean EditData;
                JOptionPane.showMessageDialog(null, "If you want to edit the data pre
                editData = JOptionPane.showInputDialog("If you want to edit the data :
              EditData= Boolean.parseBoolean(editData);
          if (EditData) {
              String fil=filename+".txt";
              String oldName = JOptionPane.showInputDialog("Type previous name ");

              String newName = JOptionPane.showInputDialog("Type new name ");
```

Message

Now your balance after chrges of $2 is $-102.00

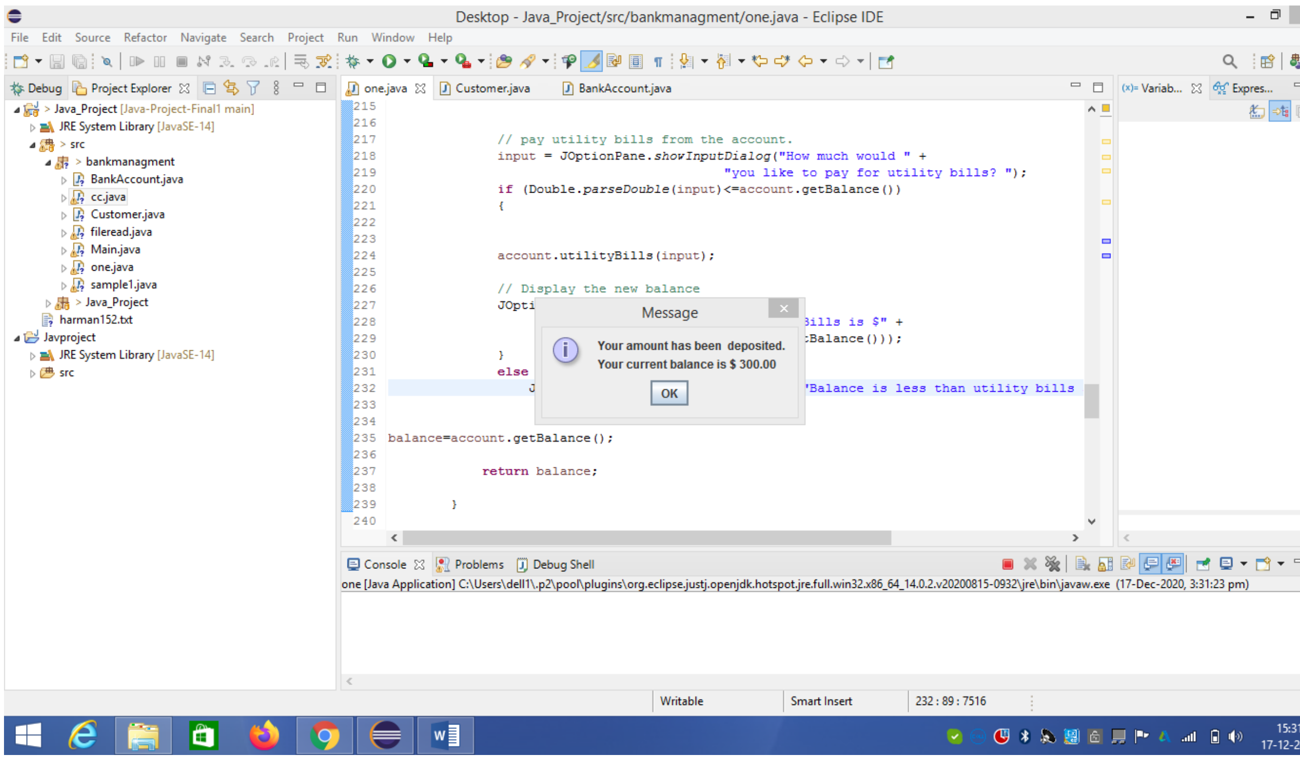
OK

```
                {
                    String data = firstFileRead1.nextLine();
                    System.out.println(data);
                }
```

one [Java Application] C:\Users\dell1\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (17-Dec-2020, 3:19:01 pm)

To overcome this error conditional loop is used
Which check if amount entered is less than or equal to the balance .
Else show the message that balance is less than the amount entered

○ **Some screenshots showing the system is working without**



Desktop - Java_Project/src/bankmanagment/one.java - Eclipse IDE

```


        // pay utility bills from the account.
        input = JOptionPane.showInputDialog("How much would " +
                "you like to pay for utility bills? ");
        if (Double.parseDouble(input)<=account.getBalance())
        {


        account.utilityBills(input);

        // Display the new balance
        JOpti                                          Bills is $" +
                                                       Balance()));

        }
        else
            J                                          "Balance is less than utility bills



balance=account.getBalance();

        return balance;

    }

```

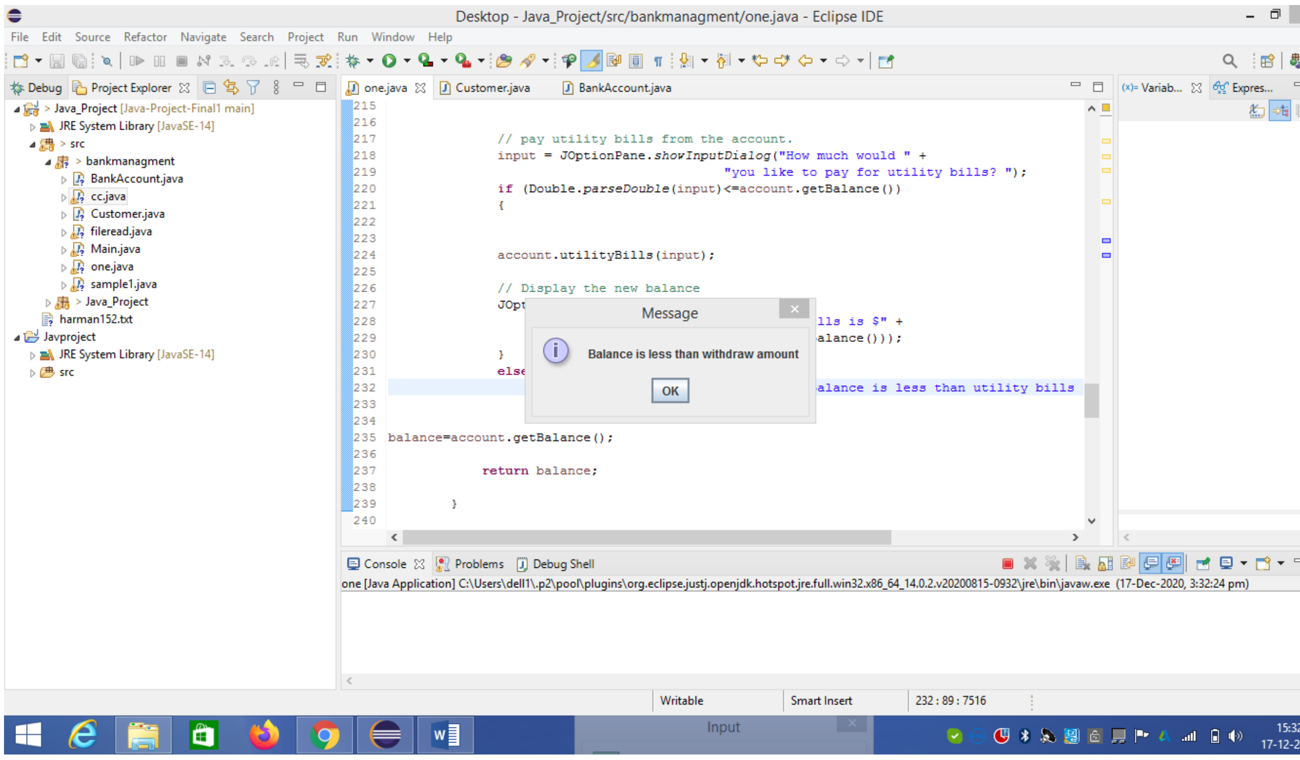
Message

Your amount has been deposited.
Your current balance is $ 300.00

OK

one [Java Application] C:\Users\dell1\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (17-Dec-2020, 3:31:23 pm)

Writable | Smart Insert | 232 : 89 : 7516



Desktop - Java_Project/src/bankmanagment/one.java - Eclipse IDE

```


        // pay utility bills from the account.
        input = JOptionPane.showInputDialog("How much would " +
                "you like to pay for utility bills? ");
        if (Double.parseDouble(input)<=account.getBalance())
        {


        account.utilityBills(input);

        // Display the new balance
        JOpt                                              lls is $" +
                                                          alance()));

        }
        else
                                                          alance is less than utility bills



balance=account.getBalance();

        return balance;

    }

```

Message

Balance is less than withdraw amount

OK

one [Java Application] C:\Users\dell1\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (17-Dec-2020, 3:32:24 pm)

Writable | Smart Insert | 232 : 89 : 7516

○ Name(s) of the file(s) the system is creating, and copy of their details

The name of the file is account number which is created by random function.
Example.   145453.txt

Following are the details showing

```
ACCOUNT NO is= 422625
   CUSTOMER Name = Harmanjeet Singh
Balance is  296.0
 interest rate is  2
Yearly charges are 200
                                   Withdrawl charges are $2 everytime
```

○ List of references

1.Slides shared by the teacher.

2. Programs shared by the teacher

3. for the buffer method we used the internet search and used following sources

- https://www.javatpoint.com/java-bufferedreader-class
- https://javaconceptoftheday.com/modify-replace-specific-string-in-text-file-in-java/#:~:text=Step%201%20%3A%20Create%20a%20File,the%20file%20to%20be%20modified.&text=Step%202%20%3A%20Initialize%20oldContent%20with,of%20the%20input%20text%20file.&text=Step%203%20%3A%20Create%20BufferedReader%20object,text%20file%20line%20by%20line.