SUBJECT NAME: PROGRAMMING IN JAVA

SUBJECT CODE: MAD-3463

TOPIC: SIMULATION OF A PRIMITIVE BANK SYSTEM

TEAM MEMBERS: ADITYA SHAW C0765770

ROOHI SINGH C0763590

PARIMAL PATEL C0764919

INDEX

INTRODUCTION

Every single one of us have a bank account nowadays, but ever thought how the process actually works? The process which might look simple is completely the opposite altogether. Yet java, provides us the opportunity to create the miniature level of a Bank.

The program in the project depicts a Banking system filled with details. The details are stored in a text file which is used to preserve the details with full security. The program serves close to being an original bank with the essential details of a customer required to open an account or access the money they have in the account.

The program serves to help the deposit, transfer and withdraw of money which makes it similar to the usual banks that we encounter in the real world.

FULL DESCRIPTION OF THE BANK

The bank presented in the program decodes the functionality of a real world bank. The bank in the program serves to execute the functionalities of the bank that we all encounter or have accounts in. The program includes a storage space to save the personal details of the customers and allows us the advantage to read and write the details from the file. The storage presented in the program is a text document i.e. database.txt.

The text file contains a set of information that is stored through array list. The account declared in the program is serve to be both of Savings and Credit also known as Chequeing account.

The program features certain characteristics and allows the customer to:

1) Search for an existing customer who has a savings or credit account
2) Deposit money
3) Withdraw money
4) Transfer money from one person to another
5) Creating a new account
6) Paying the utility bills

Thus, the virtual bank designed in the program can be compared and visualized as the bank we all are engaged with. The advantage of the code is to present the process of the bank in a miniature level where it allows changing details and storing the new information every time.

EXPLAINING THE CLASSES AND METHODS

A BRIEF DESCRIPTION OF THE CLASSES USED IN THE PROGRAM:

SERIALIZABLE: Java serialization is used in the program which is used to convert the objects in the program to byte stream. It helps in reading and writing the information presented in the code.

CLASS BANK: The class Bank acts as the super class in the program which is used to store the attribute and share it to its sub classes, to avoid redundancy of attributes, reduces complexity and saves time.

CLASS SAVING: The class "Saving" is the first sub class which inherits the attributes from the super class (Bank). This class represents the actual savings account of the customers.

CLASS CREDIT: The class "Credit" is the first sub class which inherits the attributes from the super class (Bank). This class represents the actual Chequing account of the customers.

CLASS MAIN: This is the main class where array list are created along with the detailed information of the customers in the bank. The transaction like deposit, withdraw and others are performed in this class. Reading and writing of the text file is also done in this class. Thus, this class serves as the backbone of the code.

A BRIEF DESCRIPTION OF THE METHODS USED IN THE PROGRAM:

THE PARAMERTIZED CONSTRUCTORS USED IN THE PROGRAM AS IT RESEMBLES THE INSTANCE OF A METHOD:

BANK(): Used to store the parameters defined in the class.

SAVING(): Used to store the parameters defined in the class.

CREDIT(): Used to store the parameters defined in the class.

Details_Of_All_User(): This method, as depicted from the name itself is used to store the details of all the customers who have a Savings or a Credit account with the help of array lists and is also the place to read the details in the text file.

displayAccountBasedOnUserID(): This method is used to display an existing account from the database of customers stored in the text file. The account is to be searched with the help of user id of the customer and it displays the information only if the user id matches the id presented in the database.

depositMoneyInAccount(): This method as can be understood by the name is used to deposit money in any of the accounts (Saving and Credit). The money to be deposited is an user input and the account is searched on the basis of user id of the customer. The money after depositing is added to the account of the customer.

withdrawMoneyFromAccount(): The method is used to withdraw money from the customer's account either from savings or credit. The money is thus subtracted from the initial amount only if the amount is less than the amount stored in the account.

transferMoneyToOtherAccount(): This method is used to transfer money from one account to another. The account number is to be provided from which the money id to be transferred and to which the money is getting transferred and thus the calculation takes place accordingly.

addAccount(): The method is used to provide the opportunity to new customers to add an account in the bank. It provides a choice whether to add a Saving or a Credit account with the help of a switch case.

addSaving(): This method is used to add a savings account to the bank in the same time it searches whether the account is already present or not.

addCredit(): This method is used to add a Credit account to the bank in the same time it searches whether the account is already present or not.

Populate_Data(): The method is used to fill up the details of customers in the array list and store them in the text file acting as the database.

Save_Data(): The method as understood from the name is used to save the data in the database that is in the text file by writing the new details and storing them in the database.

THE CODE EXPLAINED

The code in the system is used to create and manipulate bank account which serves to give a brief idea of how banks operate in real life.

The system here, consists of classes of different bank account followed by a way to print each of them. This is done with the help of parameterized constructors which are used to call attributes from the super (head) class, thus reducing the effort to type things again and again. The sub classes works in such a way that they contain same parameters followed by different information.

Then moving to the main class, we insert or rather store certain account details of customers from both Saving and credit account. The customer details which are their user id, name, account number, mobile number, bank name, city name are stored in a text file created through Serialization.

The program also offers the options of depositing, transferring, searching, transferring and adding account, which are some of the basic operations that a bank offers to its clients. But every time is run through nay of the mentioned operations it gives a check either throughout the user id or account number, just to make sure that data redundancy is taken care of. All these activities are executed with the help of others methods and the most important of them all was used to save the data and back to the database once the changes have been made.

Thus, in simple words the system helps a user to formulate the functions easily as it stored everything in a database though which the data can be called anytime and easily.

USER MANUAL OF THE SYSTEM

THE FOLLWING STEPS ARE FACED BY AN INDIVIDUAL IF WE TRANSFORM THE PROGRAM TO A CUSTOMER'S PERSPECTIVE

A CUSTOMER WALKS TO THE BANK CREATED IN THE SYSTEM

THE FIRST THINGS THAT IS DISPLAYED IS:

"WELCOME TO OUR BANK"

HE GOES THROUGH THE FOLLOWING OPITONS AFTER THAT IN THE MENU:

"********MENU**********"**

1. **ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED**
2. **DEPOSIT MONEY IN ACCOUNT**
3. **WITHDRAW MONEY FROM ACCOUNT**
4. **TRANSFER MONEY TO OTHER ACCOUNT**
5. **CREATE AN ACCOUNT**
6. **UTILITY BILLS**

THE CUSTOMER IS GIVEN THE LIABILITY TO CHOOSE AN OPTION:

SELECTING OPTION (1):

THE SYSTEM ASKS THE CUSTOMER TO PROVIDE THE USER ID OF THE CUSTOMER WHOSE DETAILS IS TO BE SEARCHED. ENTERING THE USER ID MAKES THE SYSTEM SEARCH FROM THE USER ID IN THE DATABASE, AND ONCE IT LOCATES THE ID, IT PRINTS ALL THE RELATED DETAILS FROM THE DATABASE. THE SYSTEM ALSO PRINTS OTHERWISE IF THE USER ID IS NOT PRESENT IN THE DATABASE.

SELECTING OPTION (2):

THE SYSTEM ALLOWS THE CUSTOMER TO DEPOSIT MONEY TO HIS ACCOUNT IN THIS OPTION. IN ORDER TO DEPOSIT MONEY THE SYSTEM ASKS THE CUSTOMER TO GIVE THE USER ID WHOSE ACCOUNT IS TO BE DEPOSITED WITH MONEY. THIS OPITON PROVIDES US WITH THE OPITON FOR BOTH SAVING AND CREDIT ACCOUNT. ONCE THE DEPOSITTED AMOUNT IS PROVIDED TO THE SYSTEM, IT AUTOMATICALLY ADDS THE AMOUNT TO THE EXISTING BALANCE.

SELECTING OPTION (3):

THE SYSTEM ALLOWS THE CUSTOMER TO WITHDRAW MONEY FROM HIS ACCOUNT IN THIS OPTION. IN ORDER TO WITHDRAW MONEY THE SYSTEM ASKS THE CUSTOMER TO GIVE THE USER ID FROM WHICH THE SYSTEM IDENTIFIES THE ACCOUNT AND ALOWS THE CUSTOMER TO WITHDRAW THE MONEY.THIS OPITON PROVIDES US WITH THE OPITON FOR BOTH SAVING AND CREDIT ACCOUNT. ONCE THE WITHDRAWED AMOUNT IS PROVIDED TO THE SYSTEM, IT AUTOMATICALLY SUBSTRACTS THE AMOUNT FROM THE EXISTING BALANCE. THE SYSTEM ALSO CHECKS WHETHER THE AMOUNT THAT IS TO BE WITHDRAWED IS LESS THAN THE EXISTING AMOUNT ELSE IT DOES NOT PROCESS.

SELECTING OPTION (4):

THIS OPTION ALLOWS THE USER TO TRANSFER MONEY FROM ONE ACCOUNT TO ANOTHER. CHOOSING THIS OPITON MAKES THE SYSTEM ASK THE CUSTOMER TO GIVE THE ACCOUNT NUMBERS FROM WHICH THE MONEY IS TO BE TRANSFERRED TO WHICH IT IS TO BE TRANSFERRED. THEN SYSTEM ASKS FOR THE AMOUNT TO BE TRANSFERRED AND MAKES SURE THAT THE ACCOUNT IS NOT MORE THAN THE AMOUNT STORED IN THE ACCOUNT FROM WHICH THE WITHDRAWL IS TAKING PLACE. THUS, ENSURING THAT THE TRANSACTION IS AS PERFECT AS POSSIBLE.

SELECTING OPTION (5):

THIS OPTION ALLOWS A NEW USER TO CREATE AN ACCOUNT IN THIS BANK. THE IS

CUSTOMER IS ALLOWED TO PICK THE TYPE OF ACCOUNT TO BE CREATED. THE OPITONS ARE SAVING AND CREDIT. THE SYSTEM MAKES SURE THAT THE ACCOUNT NUMBER SELECTED BY THE NEW CUSTOMER IS NOT A DUPLLICATED ACCOUNT AND THEN TAKES THE DETAILS REQUIRED TO CREATE A NEW ACCOUNT.

SELECTING OPTION (6):

THIS OPITON IN THE SYSTEM IS USED TO PAY THE UTILITY BILS OF THE CUSTOMERS. BILLS SUCH AS ELECTRICITY, HYDRO, MOBILE AND WIFI. THESE ARE THE MOST COMMON BILLS THAT ANY CUSTOMER WOULD LIKE TO PAY WITH A CLICK. THUS, THE SYSTEM ASKS THE CUSTOMER TO CHOOSE THE BILLS HE WOULD LIKE TO PAY AND THEN WITHDRAWS THE MONEY FROM THE ACCOUNT. THE ACCOUNT IS RECOGNISED FROM THE USER ID WHICH IS TO BE SEARCHED BY THE CUSTOMER.

SOURCE CODE OF THE PROGRAM

Bank.java

```
package bank;


import java.io.Serializable;
//Defining an abstract class and implementing it in Bank class
public class Bank implements Serializable{
                    private int account_number ;
                    private int user_id;
                    private String bank_name ;
                    private String accountholderFirstName ;
                    private String accountholderLastName ;
                    private String accountholderFullName ;
                    private int mobile_number;
                    private double bank_balance;
                    private String City_name;

                    //Creating Parameterized Constructor
                    public Bank(int account_number,int user_id,String
bank_name,String accountholderFullName,int mobile_number,double
bank_balance,String City_name)
                    {
                           super();
                           splitFullName(accountholderFullName);
                           this.user_id=user_id;
                           this.account_number = account_number;
                           this.bank_name = bank_name;
                           this.accountholderFullName = accountholderFullName;
                           this.mobile_number=mobile_number;
                           this.bank_balance=bank_balance;
                           this.City_name=City_name;
                    }
                    //Calling a method to split the name
                    private void splitFullName(String fullName) {
                           String[] splittedName = fullName.split("\\s+");
                           setaccountholderFirstName(splittedName[0]);

       setaccountholderLastName(splittedName[splittedName.length - 1]);
                    }
                    //To print the parameters
                    @Override
                    public String toString() {
                           return " account_number=" + account_number + "User_ID"
+user_id+ ", bank_name=" + bank_name + ", accountholderFirstName=" +
accountholderFirstName
                                         + ", accountholderLastName=" +
accountholderLastName + ", accountholderFullName=" + accountholderFullName+
"Mobile_number" +mobile_number+ "Bank_balance"
                                         +bank_balance+"City_name"+City_name;
                    }
                    //Getters and Setters for the parameters
                    public int getaccount_number() {
```

```
		return account_number;
	}

	public void setaccount_number(int account_number) {
		this.account_number = account_number;
	}

	public String getbank_name() {
		return bank_name;
	}

	public void setbank_name(String bank_name) {
		this.bank_name = bank_name;
	}

	public String getaccountholderFirstName() {
		return accountholderFirstName;
	}

	public void setaccountholderFirstName(String
accountholderFirstName) {
		this.accountholderFirstName = accountholderFirstName;
	}

	public String getaccountholderLastName() {
		return accountholderLastName;
	}

	public void setaccountholderLastName(String
accountholderLastName) {
		this.accountholderLastName = accountholderLastName;
	}

	public String getaccountholderFullName() {
		return accountholderFullName;
	}

	public void setaccountholderFullName(String
accountholderFullName) {
		this.accountholderFullName = accountholderFullName;
	}
	public void setmobile_number(int mobile_number)
	{
		 this.mobile_number=mobile_number;
	}
	public int getmobile_number()
	{
		return mobile_number;
	}
	public void setbank_balance(double new_balance)
	{
		 this.bank_balance=new_balance;
	}
	public double getbank_balance()
	{
		return bank_balance;
	}
	public void setCity_name(String City_name)
	{
```

```
            this.City_name=City_name;
        }
        public String getCity_name()
        {
            return City_name;
        }

        public int getUser_id() {
            return user_id;
        }

        public void setUser_id(int user_id) {
            this.user_id = user_id;
        }


}
```

Saving.java

```
package bank;

public class Saving extends Bank{

    private static double interestRate;

     Saving(int account_number,int user_id,String bank_name,String
accountholderFullName,int mobile_number,double bank_balance,String City_name)
    {

super(account_number,user_id,bank_name,accountholderFullName,mobile_number,bank_ba
lance,City_name);
        interestRate=5;

    }
     @Override
      public String toString() {
       return " Saving account Details:\n"
                                  + " Account_Number - "+this.getaccount_number()
+"\n"
                                  + " User ID - "+this.getUser_id()+"\n"
                                  + " Client Name -
"+this.getaccountholderFullName()+"\n"
                                  + " Mobile Number -
"+this.getmobile_number()+"\n"
                                  + " Balance - "+this.getbank_balance()+"\n"
                                  + " City - "+this.getCity_name()+"\n";
      }
                                            }
```

Credit.java

```
package bank;

public class Credit extends Bank{


        //Constructor
         Credit(int account_number,int user_id,String bank_name,String
accountholderFullName,int mobile_number,double bank_balance,String City_name) {

super(account_number,user_id,bank_name,accountholderFullName,mobile_number,bank_ba
lance,City_name);}


                    @Override
                        public String toString() {
                              return "Credit account Details:\n"
                                          + " Account_Number -
"+this.getaccount_number() +"\n"
                                          + " User ID -
"+this.getUser_id()+"\n"
                                          + " Client Name -
"+this.getaccountholderFullName()+"\n"
                                          + " Mobile Number -
"+this.getmobile_number()+"\n"
                                          + " Balance -
"+this.getbank_balance()+"\n"
                                          + " City -
"+this.getCity_name()+"\n";
                                                 }}
```

Main.java

```
package bank;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;
public class Main {



                    //Create Global ArrayLists for Savings and Credits
                    private static ArrayList<Saving> Savings_Acc;
                    private static ArrayList<Credit> Credit_Acc;
                    private static ArrayList<Bank> Total_details; //Contains both
Savings and Credits


                    private static Scanner sc;
                    public static void main(String[] args) throws
ClassNotFoundException{

                          Details_Of_All_User();
                           sc = new Scanner (System.in);


                          println("WELCOME TO OUR BANK");

                          try {
                                int choice = 1 ;
                                while (choice > 0 && choice < 6) {
                                      System.out.println("----------------------
------------------------------");

      println("**********************MENU*******************************");
                                      System.out.println("----------------------
------------------------------");
                                      println("Please Select your Option :-");
                                      println("(1.) ENTER THE USER ID OF THE
PERSON WHOSE ACCOUNT IS TO BE SEARCHED");
                                      println("(2.) DEPOSIT MONEY IN ACCOUNT");
                                      println("(3.) WITHDRAW MONEY FROM
ACCOUNT");
                                      println("(4.) TRANSFER MONEY TO OTHER
ACCOUNT");
                                      println("(5.) CREATE AN ACCOUNT");
                                      println("(6.) PAY UTILITY BILLS");
                                      println("Enter any other Number to Exit :
");
                                      choice = sc.nextInt();

                                      switch(choice) {
```

```
                                        case 1:
displayAccountBasedOnUserID();
                                              break;
                                        case 2:
                                              depositMoneyInAccount();
                                              break;
                                        case 3:
                                              withdrawMoneyFromAccount();
                                              break;
                                        case 4:
transferMoneyToOtherAccount();
                                              break;
                                        case 5:
                                              addAccount();
                                              break;
                                        case 6:
                                              payutilitybills();
                                        default:
                                              println("");
                                              println("Thankyou for using
our Publications App. Bye!");
                                              break;
                                  }
                            }
                      } catch (InputMismatchException e) {
                            println("");
                                  println("Please Enter Valid Input (Input
Mismatch Exception)");
                            println("");
                      } finally {
                            sc.close();
                            println("Program Terminated");
                      }

               }

               private static void print(String s) {
                      System.out.print(s);

               }

               private static void println(String s) {
                      System.out.println(s);
               }

               private static void Details_Of_All_User() throws
ClassNotFoundException {


                      try {
                            FileInputStream fi = new FileInputStream(new
File("C:\\Database\\Database.txt"));
                            ObjectInputStream oi = new ObjectInputStream(fi);

                            // Read objects
                            Savings_Acc = (ArrayList<Saving>)
oi.readObject();
```

```
                                Credit_Acc = (ArrayList<Credit>) oi.readObject();
                                Total_details = (ArrayList<Bank>)
oi.readObject();
                                oi.close();
                                fi.close();
                        } catch (FileNotFoundException e) {
                                System.out.println("File not found");
                                Populate_Data();
                        } catch (IOException e) {
                                System.out.println("Error initializing stream" +
e);
                        }

                }

                //Displaying the account details after searching
                private static void displayAccountBasedOnUserID() {
                        println("");
                        println("Enter the User_ID of the account holder");
                        int user_id = sc.nextInt();
                        boolean found = false ;

                        for(Bank Saving : Savings_Acc) {
                                if(user_id==Saving.getUser_id()){
                                        println(""+Saving.toString());
                                        found = true;
                                }
                        }
                        for(Bank Credit : Credit_Acc) {
                                if(user_id==Credit.getUser_id()){
                                        println("----------------------------------
------------------");
                                        println("THE ACCOUNT HAS BEEN FOUND ");
                                        println("----------------------------------
------------------");
                                        println(""+Credit.toString());
                                        found = true;
                                }
                        }

                        //If not found
                        if(!found){
                                println("");
                                println("None of the Savings or Credit Account
have this  user id \'"+user_id+"\'");
                        }
                        println("");
                }

                //Deposit the money
                private static void depositMoneyInAccount() {
                        println("");
                        double New_balance = 0 ;
                        print("Enter UserID : ");
                        int user_id = sc.nextInt();
                        for(Bank Saving: Savings_Acc) {
                                if(user_id==Saving.getUser_id()){
                                        println("----------------------------------
------------------");
```

```
                                        print("Do you want to Deposit in Savings
Account : Y/N ?");
                                        println("-------------------------------
------------------");
                                        String opt = sc.next();
                                        if(opt.equalsIgnoreCase("Y")) {
                                              System.out.println("Enter The Amount
you Want To Deposit into Saving Account :");
                                              double amount = sc.nextInt();
                                              New_balance =
Saving.getbank_balance() + amount;
                                              Saving.setbank_balance(New_balance);
                                              println("Balance :"
+(Saving.getbank_balance()));
                                        }
                                  }
                            }

                            for(Bank Credit : Credit_Acc) {
                                  if(user_id==Credit.getUser_id()){
                                        println("-------------------------------
------------------");
                                        print("Deposit in Credit Account : Y/N
?");
                                        println("-------------------------------
------------------");
                                        String opt = sc.next();
                                        if(opt.equalsIgnoreCase("Y")) {
                                        System.out.println("Enter The Amount you
Want To Deposit into Credit Account :");
                                        double amount = sc.nextInt();
                                        New_balance = Credit.getbank_balance() +
amount;
                                        Credit.setbank_balance(New_balance);
                                        println("Balance :"
+(Credit.getbank_balance()));
                                        }
                                  }

                            }

                            Save_Data();
                            println("");
                      }

                      //Withdrawing money from account
                      private static void withdrawMoneyFromAccount() {
                            println("");
                            double New_balance = 0 ;
                            print("Enter UserID : ");
                            int user_id = sc.nextInt();
                            for(Bank Saving: Savings_Acc) {
                                  if(user_id==Saving.getUser_id()){
                                        println("-------------------------------
------------------");
                                        print("Withdraw from Savings Account : Y/N
?");
                                        println("-------------------------------
------------------");
```

```
                                        String opt = sc.next();
                                        if(opt.equalsIgnoreCase("Y")) {
                                              System.out.println("Enter The Amount
you Want To Withdraw from Saving Account :");
                                              double amount = sc.nextInt();
                                              if(amount>Saving.getbank_balance())
{
                                                    println("Balance less than
amount entered");
                                              } else {
                                                    New_balance =
Saving.getbank_balance() - amount;

      Saving.setbank_balance(New_balance);
                                              }
                                              println("Balance :"
+(Saving.getbank_balance()));
                                        }
                                  }
                            }

                            for(Bank Credit : Credit_Acc) {
                                  if(user_id==Credit.getUser_id()){
                                        println("----------------------------------
------------------");
                                        print("Withdraw from Credit Account : Y/N
?");
                                        println("----------------------------------
------------------");
                                        String opt = sc.next();
                                        if(opt.equalsIgnoreCase("Y")) {
                                        System.out.println("Enter The Amount you
Want To Withdraw from Credit Account :");
                                        double amount = sc.nextInt();
                                        if(amount>Credit.getbank_balance()) {
                                              println("Balance less than amount
entered");
                                        } else {
                                              New_balance =
Credit.getbank_balance() - amount;
                                              Credit.setbank_balance(New_balance);
                                        }
                                        println("Balance :"
+(Credit.getbank_balance()));
                                        }
                                  }

                            }

                            Save_Data();
                            println("");
                      }

                      //Transferring money to other Account
                      private static void transferMoneyToOtherAccount() {
                            Boolean found1 =false, found2 = false, lowbal = false;
                            println("");
                            println("Enter the Account number from which transfer
to be made:-");
```

```
                        int from = sc.nextInt();
                        println("Enter the Account number to which transfer to
be made:-");
                        int to = sc.nextInt();
                        println("Enter amount to be transferred");
                        int amount = sc.nextInt();
                        for(Bank Saving : Savings_Acc) {
                                if(from==Saving.getaccount_number()){
                                        if(amount > Saving.getbank_balance()) {
                                                println("Low Balance !");
                                                lowbal= true;
                                        }else {

        Saving.setbank_balance(Saving.getbank_balance()-amount);
                                                found1 = true;
                                        }
                                }
                        }
                        for(Bank Credit : Credit_Acc) {
                                if(from==Credit.getaccount_number()){
                                        if(amount > Credit.getbank_balance()) {
                                                println("Low Balance !");
                                                lowbal = true;
                                        }else {

        Credit.setbank_balance(Credit.getbank_balance()-amount);
                                                found1 = true;
                                        }
                                }
                        }

                        for(Bank Saving : Savings_Acc) {
                                if(to==Saving.getaccount_number()){
                                        found2 = true;

        Saving.setbank_balance(Saving.getbank_balance()+amount);
                                }
                        }
                        for(Bank Credit : Credit_Acc) {
                                if(to==Credit.getaccount_number()){
                                        found2 = true;

        Credit.setbank_balance(Credit.getbank_balance()+amount);
                                }
                        }
                        if(found1 && found2 && !lowbal)
                        {
                                println("Transaction Succesfull");
                                Save_Data();
                        }else if(!found1 && !found2) {
                                println("Bank account(s) not found");
                        }
                }

                //Add Savings and Credit
                private static void addAccount() {
                        println("");
                        println("1. Add Saving");
                        println("2. Add Credit");
```

```
println("Enter your Choice: ");
int choice = sc.nextInt();

if (choice == 1 ){
	addSaving();
} else if (choice == 2) {
	addCredit();
} else {
	println("Please Enter Valid Input");
}
Save_Data();
}

//Adding a Savings account
private static void addSaving() {

	Scanner sc1 = new Scanner(System.in);

	Scanner sc2 = new Scanner(System.in);
	Boolean conflict= false;
	println("");
	int account_number ;
	while(true) {
	println("Enter the account number : ");
	 account_number = sc1.nextInt() ;
		for(Saving saving: Savings_Acc) {

	if(saving.getaccount_number()==account_number) {
					conflict=true;
				}
			}
			for(Credit credit: Credit_Acc) {

	if(credit.getaccount_number()==account_number) {
					conflict=true;
				}
			}
			if(!conflict) {
				break;
			}
		}
		println("Enter the User ID : ");
		int user_Id = sc1.nextInt() ;
		println("Enter the bank name : ");
		String bank_name = sc1.next();
		println("Enter the account holder's full name : ");
		String accountholderFullName = sc2.nextLine();
		println("Enter mobile number of the account holder :
");
		int mobile_number= sc1.nextInt();
		println("Enter the bank balance : ");
		double bank_balance = sc1.nextDouble();
		println("Enter the City name : ");
		String City_name=sc2.nextLine();


		Saving newSaving = new Saving(account_number,user_Id,
bank_name, accountholderFullName, mobile_number,bank_balance,City_name);
		Savings_Acc.add(newSaving);
```

```
                        Total_details.add(newSaving);
                        println("");
                        println("Saving Added! "+newSaving.toString());
                        println("");
                }

                //Adding a Credit account
                private static void addCredit() {

                        Scanner sc1 = new Scanner(System.in);

                        Scanner sc2 = new Scanner(System.in);
                        Boolean conflict = false;
                        int account_number;
                        println("");
                        while(true) {
                                println("Enter the account number : ");
                                account_number = sc1.nextInt() ;
                                        for(Credit credit: Credit_Acc) {

        if(credit.getaccount_number()==account_number) {
                                                        conflict=true;
                                                }
                                        }
                                        for(Saving saving: Savings_Acc) {

        if(saving.getaccount_number()==account_number) {
                                                        conflict=true;
                                                }
                                        }
                                        if(!conflict) {
                                                break;
                                        }
                                }
                        println("Enter the User ID : ");
                        int user_Id = sc1.nextInt() ;
                        println("Enter the bank name : ");
                        String bank_name = sc1.next();
                        println("Enter the account holder's full name : ");
                        String accountholderFullName = sc2.nextLine();
                        println("Enter mobile number of the account holder :
");
                        int mobile_number= sc1.nextInt();
                        println("Enter the bank balance : ");
                        double bank_balance = sc1.nextDouble();
                        println("Enter the City name : ");
                        String City_name=sc2.nextLine();


                        Credit newCredit = new Credit(account_number,user_Id,
bank_name, accountholderFullName, mobile_number,bank_balance,City_name);
                        Credit_Acc.add(newCredit);
                        Total_details.add(newCredit);
                        println("");
                        println("Credit Added! "+newCredit.toString());
                        println("");
                }
                private static void payutilitybills() {
                        println("");
```

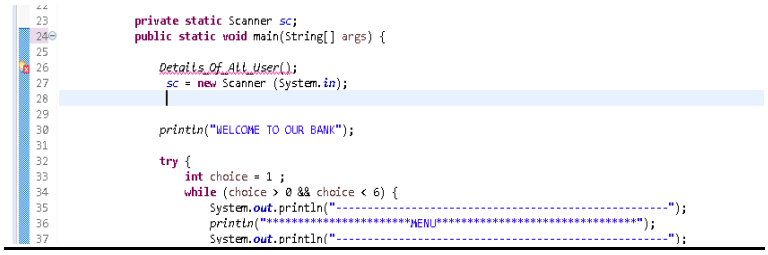
```
                        double New_balance = 0 ;
                        print("Enter UserID : ");
                        int user_id = sc.nextInt();
                        for(Bank Credit: Credit_Acc) {
                              if(user_id==Credit.getUser_id()){
                                    println("----------------------------------
------------------");
                                    println("Would you like to pay your bills
: Y/N ?");
                                    println("----------------------------------
------------------");
                                    String opt = sc.next();
                                    if(opt.equalsIgnoreCase("Y")) {
                                          System.out.println("Total bill is
$230 which is divided :");
                                          System.out.println("ELECTRICITY:
$60");
                                          System.out.println("HYDRO: $20");
                                          System.out.println("MOBILE: $90");
                                          System.out.println("WIFI: $120");
                                          int amount= 290;
                                          if(amount>Credit.getbank_balance())
{
                                                println("Balance less than
amount entered");
                                                Credit.setbank_balance(0);
                                          } else {
                                                New_balance =
Credit.getbank_balance() - amount;

      Credit.setbank_balance(New_balance);
                                          }
                                          println("Balance :"
+(Credit.getbank_balance()));
                                    }
                              }
                        }

                        Save_Data();
                        println("");
                  }

                  private static void Populate_Data() {
                        //Initialize Savings
                        Saving Saving1 = new Saving(2001201,001,"ICICI","Aditya
Shaw",76542110,1000.50,"toronto");
                        Saving Saving2 = new
Saving(2001202,002,"ICICI","Parimal Patel",76542220,1000.50,"toronto");
                        Saving Saving3 = new Saving(2001203,003,"ICICI","John
Miller",76542330,1000.50,"toronto");
                        Saving Saving4 = new
Saving(2001204,004,"ICICI","Prithvi Shaw",76542440,1000.50,"toronto");
                        Saving Saving5 = new Saving(2001205,005,"ICICI","Adam
Gill",76542550,1000.50,"toronto");

                        //Initialize Credits
                        Credit Credit1 = new Credit(2001206,105,"ICICI","Mark
Dcosta",76542550,1000.50,"toronto");
```

```
                        Credit Credit2 = new Credit(2001207,001,"ICICI","Aditya
Shaw",76542110,1000.50,"toronto");
                        Credit Credit3 = new
Credit(2001208,002,"ICICI","Parimal PAtel",76542220,1000.50,"toronto");
                        Credit Credit4 = new
Credit(2001209,103,"ICICI","Prithvi Shaw",98765330,1000.50,"toronto");
                        Credit Credit5 = new Credit(2001210,104,"ICICI","Sachin
Dev",76542440,1000.50,"toronto");

                        //Initialize ArrayLists
                        Savings_Acc = new ArrayList<>();
                        Credit_Acc = new ArrayList<>();
                        Total_details = new ArrayList<>();

                        //Add all Savings to Savings_Acc
                        Savings_Acc.add(Saving1);
                        Savings_Acc.add(Saving2);
                        Savings_Acc.add(Saving3);
                        Savings_Acc.add(Saving4);
                        Savings_Acc.add(Saving5);

                        //Add all Credits to Credit_Acc
                        Credit_Acc.add(Credit1);
                        Credit_Acc.add(Credit2);
                        Credit_Acc.add(Credit3);
                        Credit_Acc.add(Credit4);
                        Credit_Acc.add(Credit5);

                        //Add all Savings in Total Details
                        Total_details.addAll(Savings_Acc);
                        Total_details.addAll(Credit_Acc);
                        try {
                                FileOutputStream f = new FileOutputStream(new
File("C:\\Database\\Database.txt"));
                                ObjectOutputStream o = new ObjectOutputStream(f);

                                // Write objects to file
                                o.writeObject(Savings_Acc);
                                o.writeObject(Credit_Acc);
                                o.writeObject(Total_details);
                                o.close();
                                f.close();
                                println("----------------------------------------
------------");
                                println("New .txt Database created.");
                                println("----------------------------------------
------------");
                        } catch (FileNotFoundException e) {
                                System.out.println("File not found");
                        } catch (IOException e) {
                                System.out.println("Error initializing stream");
                        }


                }
                //Creating a method to save data in Database.txt
                private static void Save_Data() {
                        try {
```

```
                                FileOutputStream f = new FileOutputStream(new
File("C:\\Database\\Database.txt"));
                                ObjectOutputStream o = new ObjectOutputStream(f);

                                // Write objects to file
                                o.writeObject(Savings_Acc);
                                o.writeObject(Credit_Acc);
                                o.writeObject(Total_details);
                                o.close();
                                f.close();
                                println("New data saved.");
                          } catch (FileNotFoundException e) {
                                System.out.println("File not found");
                          } catch (IOException e) {
                                System.out.println("Error initializing stream");
                          }

                    }
}
```

ERRORS ENCOUNTERED DURING CODING AND EXECUTING

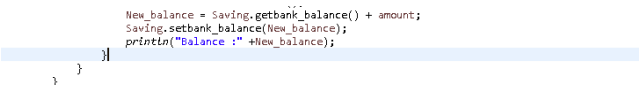
1)



```
        private static Scanner sc;
        public static void main(String[] args) {

            Details_Of_All_User();
             sc = new Scanner (System.in);

            println("WELCOME TO OUR BANK");

            try {
                int choice = 1 ;
                while (choice > 0 && choice < 6) {
                    System.out.println("-------------------------------------------------");
                    println("*********************MENU*******************************");
                    System.out.println("-------------------------------------------------");
```

The problem was faced as the method's class was not found. It was solved by adding a "`throws` ClassNotFoundException".

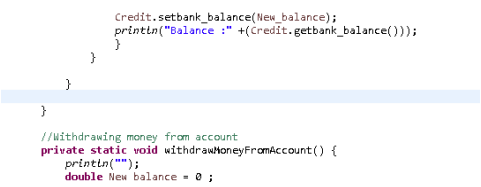
2)



```
                New_balance = Saving.getbank_balance() + amount;
                Saving.setbank_balance(New_balance);
                println("Balance :" +New_balance);
            }
        }
    }
```

The statement "println("Balance:" + `New_balance); would give the right output but it does not write the new files in the database. Thus changing the code to println("Balance :" +(Saving.getbank_balance())); helps to print the right output as well as save the new output in the database.`

3)



```
                Credit.setbank_balance(New_balance);
                println("Balance :" +(Credit.getbank_balance()));
                }
            }

        }
    }

    //Withdrawing money from account
    private static void withdrawMoneyFromAccount() {
        println("");
        double New_balance = 0 ;
```

The statement here had no problems yet the output being generated was to be stored in the memory. Thus we had to create a method that would help to write the information in the database that is our text file. Hence, we added a method called "`Save_Data()`"

SYSTEM OUTPUT

THE FILES THAT IS DECLARED TO TEST THE OPTIONS:

```
private static void Populate_Data() {
    //Initialize Savings
    Saving Saving1 = new Saving(2001201,001,"ICICI","Aditya Shaw",76542110,1000.50,"toronto");
    Saving Saving2 = new Saving(2001202,002,"ICICI","Parimal Patel",76542220,1000.50,"toronto");
    Saving Saving3 = new Saving(2001203,003,"ICICI","John Miller",76542330,1000.50,"toronto");
    Saving Saving4 = new Saving(2001204,004,"ICICI","Prithvi Shaw",76542440,1000.50,"toronto");
    Saving Saving5 = new Saving(2001205,005,"ICICI","Adam Gill",76542550,1000.50,"toronto");

    //Initialize Credits
    Credit Credit1 = new Credit(2001206,105,"ICICI","Mark Dcosta",76542550,1000.50,"toronto");
    Credit Credit2 = new Credit(2001207,001,"ICICI","Aditya Shaw",76542110,1000.50,"toronto");
    Credit Credit3 = new Credit(2001208,002,"ICICI","Parimal PAtel",76542220,1000.50,"toronto");
    Credit Credit4 = new Credit(2001209,103,"ICICI","Prithvi Shaw",98765330,1000.50,"toronto");
    Credit Credit5 = new Credit(2001210,104,"ICICI","Sachin Dev",76542440,1000.50,"toronto");

    //Initialize ArrayLists
```

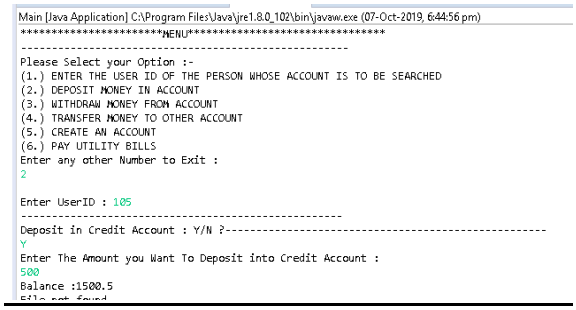
SEARCHING AN ACCOUNT BY THE USER ID

```
Main [Java Application] C:\Program Files\Java\jre1.8.0_102\bin\javaw.exe (07-Oct-2019, 6:44:56 pm)
WELCOME TO OUR BANK
-----------------------------------------------------
**********************MENU*********************************
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
1

Enter the User_ID of the account holder
105
-----------------------------------------------------
THE ACCOUNT HAS BEEN FOUND
-----------------------------------------------------
Credit account Details:
 Account_Number - 2001206
 User ID - 105
 Client Name - Mark Dcosta
 Mobile Number - 76542550
 Balance - 1000.5
 City - toronto
```

SEARCHING A USER ID HAVING BOTH ACCOUNTS

```
Problems  Javadoc  Declaration  Console
Main [Java Application] C:\Program Files\Java\jre1.8.0_102\bin\javaw.exe (07-Oct-2019, 6:44:56 pm)
-----------------------------------------------------
**********************MENU*********************************
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
1

Enter the User_ID of the account holder
001
Saving account Details:
 Account_Number - 2001201
 User ID - 1
 Client Name - Aditya Shaw
 Mobile Number - 76542110
 Balance - 1000.5
 City - toronto

-----------------------------------------------------
THE ACCOUNT HAS BEEN FOUND
-----------------------------------------------------
Credit account Details:
 Account_Number - 2001207
 User ID - 1
 Client Name - Aditya Shaw
 Mobile Number - 76542110
 Balance - 1000.5
 City - toronto
```

DEPOSITNG MONEY ND CHECKING THE UPDATE



```
Main [Java Application] C:\Program Files\Java\jre1.8.0_102\bin\javaw.exe (07-Oct-2019, 6:44:56 pm)
**********MENU**********
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
2

Enter UserID : 105
-----------------------------------------------------
Deposit in Credit Account : Y/N ?-----------------------------------------------------
Y
Enter The Amount you Want To Deposit into Credit Account :
500
Balance :1500.5
File not found
```

```
-----------------------------------------------------
**********MENU**********
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
1

Enter the User_ID of the account holder
105
-----------------------------------------------------
THE ACCOUNT HAS BEEN FOUND
-----------------------------------------------------
Credit account Details:
 Account_Number - 2001206
 User ID - 105
 Client Name - Mark Dcosta
 Mobile Number - 76542550
 Balance - 1500.5
 City - toronto
```

WITHDRAWING MONEY FROM AN ACCOUNT AN CHECKING THE DETAILS

```
**********MENU**********
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
3

Enter UserID : 105
-----------------------------------------------------
Withdraw from Credit Account : Y/N ?-----------------------------------
Y
Enter The Amount you Want To Withdraw from Credit Account :
125
Balance :1375.5
```

```
-----------------------------------------------------
**********MENU**********
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
1

Enter the User_ID of the account holder
105
-----------------------------------------------------
THE ACCOUNT HAS BEEN FOUND
-----------------------------------------------------
Credit account Details:
 Account_Number - 2001206
 User ID - 105
 Client Name - Mark Dcosta
 Mobile Number - 76542550
 Balance - 1375.5
 City - toronto
```

TRANSFERRING MONEY FROM ONE BANK TO ANOTHER

```
--------------------------------------------------
**********MENU**********
--------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
4

Enter the Account number from which transfer to be made:-
2001201
Enter the Account number to which transfer to be made:-
2001202
Enter amount to be transferred
300
Transaction Succesfull
```

```
--------------------------------------------------
**********MENU**********
--------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
1

Enter the User_ID of the account holder
001
Saving account Details:
 Account_Number - 2001201
 User ID - 1
 Client Name - Aditya Shaw
 Mobile Number - 76542110
 Balance - 700.5
 City - toronto

--------------------------------------------------
THE ACCOUNT HAS BEEN FOUND
--------------------------------------------------
```

```
--------------------------------------------------
**********MENU**********
--------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
1

Enter the User_ID of the account holder
002
 Saving account Details:
 Account_Number - 2001202
 User ID - 2
 Client Name - Parimal Patel
 Mobile Number - 76542220
 Balance - 1300.5
 City - toronto
```

ADDING ACCOUNT

```
--------------------------------------------------
**********MENU**********
--------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
5

1. Add Saving
2. Add Credit
Enter your Choice:
1

Enter the account number :
2001212
Enter the User ID :
111
Enter the bank name :
SCOTIA
Enter the account holder's full name :
BEN STOKES
Enter mobile number of the account holder :
78796544
Enter the bank balance :
750
Enter the City name :
NOVA SCOTIA
```

```
Saving Added!  Saving account Details:
 Account_Number - 2001212
 User ID - 111
 Client Name - BEN STOKES
 Mobile Number - 78796544
 Balance - 750.0
 City - NOVA SCOTIA
```

PAYING UTILITY BILLS

```
WELCOME TO OUR BANK
-----------------------------------------------------
**********************MENU*********************************
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
6

Enter UserID : 105
-----------------------------------------------------
Would you like to pay your bills : Y/N ?
-----------------------------------------------------
Y
Total bill is $230 which is divided :
ELECTRICITY: $60
HYDRO: $20
MOBILE: $90
WIFI: $120
Balance :710.5
```

```
-----------------------------------------------------
**********************MENU*********************************
-----------------------------------------------------
Please Select your Option :-
(1.) ENTER THE USER ID OF THE PERSON WHOSE ACCOUNT IS TO BE SEARCHED
(2.) DEPOSIT MONEY IN ACCOUNT
(3.) WITHDRAW MONEY FROM ACCOUNT
(4.) TRANSFER MONEY TO OTHER ACCOUNT
(5.) CREATE AN ACCOUNT
(6.) PAY UTILITY BILLS
Enter any other Number to Exit :
1

Enter the User_ID of the account holder
105
-----------------------------------------------------
THE ACCOUNT HAS BEEN FOUND
-----------------------------------------------------
Credit account Details:
 Account_Number - 2001206
 User ID - 105
 Client Name - Mark Dcosta
 Mobile Number - 76542550
 Balance - 710.5
 City - toronto
```

NAME OF THE FILES THE SYSTEM HAS CREATED ALONG WITH THEIR COPIES

The system has 4 classes named:

1) Bank.java : the super class containing attributes
2) Saving.java : the subclass and one of the account
3) Credit.java : another subclass and the other account
4) Main.java: The class where all the operations are taking place

The text files created is named as:

Database.txt : The text file containing all the details of the existing customers and the new customers that are to be created.

CONCLUSION

The creation of the multi account Banking system was the amalgamation of research and experimenting with different techniques to make this program as easy as possible. The idea of the different banking systems was taken from Google itself.

Creating the system was a challenge initially, but with advance studies from different websites, the ultimate code can out to be running smoothly. The program was kept as simple as possible by providing two types of bank account. The Savings and the Credit accounts, usually the credit account is the chequing account which is generally used to pay the utility bills.

CHALLENGES FACED WITH THEIR SOLUTIONS:

The challenges commonly faced could be divided into certain groups:

1) Initial lack of synchronisation as the code was divided among the team members. Everyone was given their part of responsibility but grouping the program together led to certain errors.
2) Understanding the methods to be made along with dividing the classes took a lot of time, this was taken care of eventually, as new improved methods came up with every error that the code was facing.
3) Researching with user defined Array list was the second toughest part of the code, we had to refer to a lot of websites and books from Toronto public library.
4) The hardest part was to create a database which could read as well as write the information in the text file that was created.

DISTRIBUTION OF WORK AMONG TEAM MEMBERS:

ADITYA SHAW: Understanding the concept of file handling in java (text file) along with reading and writing of the file.

Creating the pdf file.

PARIMAL PATEL: Forming the methods and their requirements, taking care of transfer function.

ROOHI SINGH: Understanding the concept of user defined array list and the attaching it to the text file.

THE CODE WAS A COMBINED EFFORT GIVEN BY ALL THE RESPECTED TEAM MEMBERS.

REFERENCES:

THE WEBSITES BELOW ABRIDGED THE DIFFERENCES WE HAD WITH CODES:

1) www.w3schools.com
2) www.javapoint.com
3) www.stackoverflow.com

THE BOOK NAMED "HEAD FIRST JAVA" **WAS ALSO REFERRED IN GAINING CERTAIN CONCEPTS.**