Informatics Institute of Technology
Department of Computing

Module: 4COSC0010C – Programming Principles 02

Degree Program: BScCS

Tutorial Group: Group A

Module Leader: Mr. Guganathan Poravi

Coursework 01 – Stage 4

Date of submission: 11.03.2018

Student ID - IIT Student ID: 2017091

UoW ID : 16737363/1

Student First Name: Dinithi

Student Surname : Jayasekara

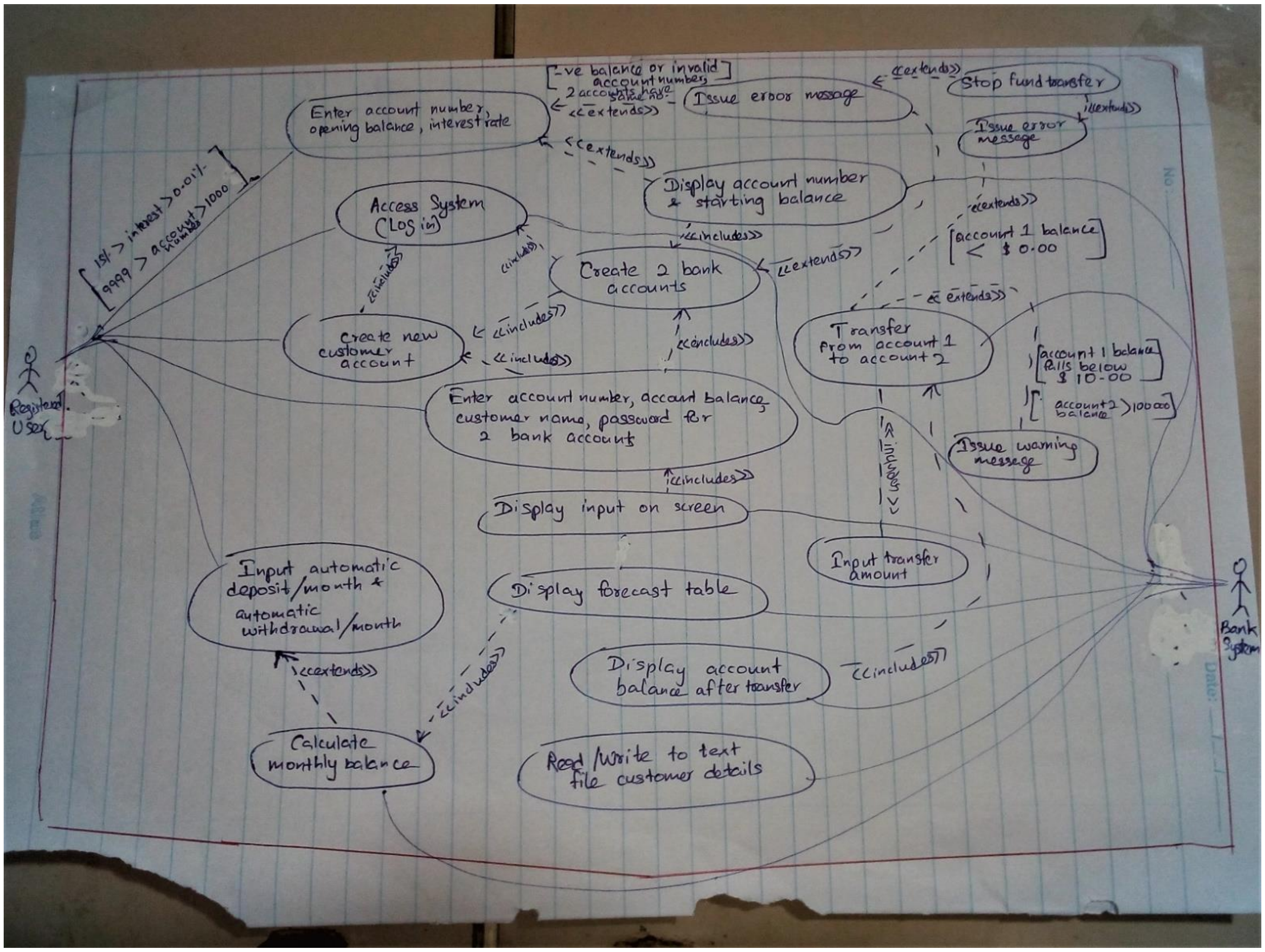
Table of Contents

1. INTRODUCTION

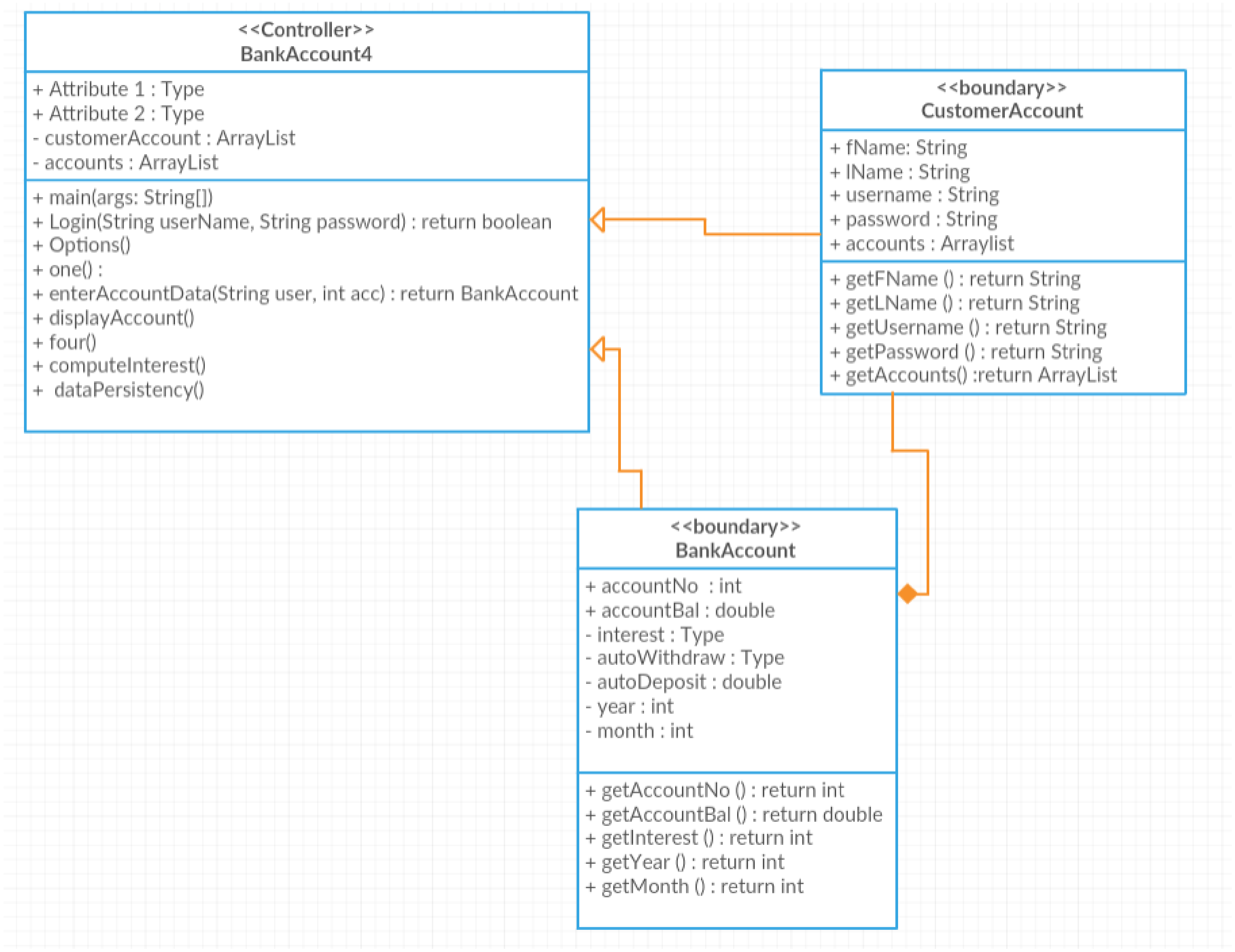
This is a JAVA console-based application developed for InterBanking Pty. The purpose of this system is to produce a next generation customer and account management system. This application has been developed for employees of the bank to add new customer accounts and bank accounts and is very user friendly serving its purpose to the fullest.

2. DESIGN

- **UML DIAGRAM:**


Registered User
Bank System
Enter account number, opening balance, interest rate
[-ve balance or invalid account numbers, 2 accounts have same no.]
<<extends>>
Issue error message
<<extends>>
Stop fund transfer
<<extends>>
Issue error message
[15% > interest > 0.01%
9999 > account number > 1000]
<<extends>>
Display account number & starting balance
<<extends>>
Access System (Log in)
<<includes>>
<<includes>>
<<includes>>
Create 2 bank accounts
<<extends>>
[account 1 balance < $ 0.00]
<<extends>>
Create new customer account
<<includes>>
<<includes>>
<<includes>>
Transfer from account 1 to account 2
[account 1 balance falls below $ 10.00]
[account2 balance > 10000]
Enter account number, account balance, customer name, password for 2 bank accounts
Issue warning message
<<includes>>
<<includes>>
Display input on screen
Input transfer amount
Input automatic deposit/month & automatic withdrawal/month
Display forecast table
<<extends>>
<<includes>>
Display account balance after transfer
<<includes>>
Calculate monthly balance
Read /Write to text file customer details

- **CLASS DIAGRAM:**



- **ALGORITHM:**

  - To declare a **BankAccount** object and continues to get **BankAccount** values from the user until the user enters a **BankAccount** with value 0 for the account number.

```
WHILE (TRUE){
    PRINT "Please enter an account number"
    PROMPT account number
    IF (account number null || !account number > 10000) {
        PRINT "Please enter a valid account number"
        CONTINUE
    } ELSE IF (account number == 0) {
        BREAK;

    } ELSE IF (account number < 1000 || account number> 9999) {
        PRINT "Account number must be between 1000 - 9999"
        CONTINUE
    } ELSE {
        CREATE new account object
        INPUT account number to new account object
        ADD account object to accounts array list
       }
}
```

3. IMPLEMENTATION

- Write an enterAccountData() method that declares a local BankAccount object and prompt the user for values for each data field. The method returns a data filled BankAccount object to the calling method.

```
public static BankAccount enterAccountData(String user, int acc) {
    BankAccount newAccount = new BankAccount ();
    newAccount.accountNo = acc;
    for (int i = 0; i < customerAccount.size (); i++) {
        if ((customerAccount.get (i).username.equalsIgnoreCase (user))) {
            customerAccount userX = com.company.BankAccount4.customerAccount.get (i);
            System.out.println ("Enter " + userX.fName + " " + userX.lName + "'s" + " account 
opening balance in dollars($): ");
            newAccount.accountBal = sc.nextDouble ();
            while (newAccount.accountBal < 0 || newAccount.accountBal > 100000) {
                System.out.println ("Invalid Account Balance! Account balance has to be within the 
range of $ 0.00 to $100000.00");
                System.out.println ("Account Opening Balance :");
                newAccount.accountBal = sc.nextDouble ();
            }
            System.out.println ("********INTEREST RATES********");
            System.out.println ("Enter annual interest rate for bank account: ");
            newAccount.interest = sc.nextDouble ();
            do {
                if (!(newAccount.interest > 0.01 && newAccount.interest < 15)) {
                    System.out.println ("Invalid annual interest rate. Please input a new value 
between the range of 0.01% to 15%");
                    System.out.println ("Please re-enter a suitable interest rate for bank account: 
");

                    newAccount.interest = sc.nextDouble ();
                }
            } while ((!(newAccount.interest > 0.01 && newAccount.interest < 15)));
            newAccount.accountBal = (newAccount.accountBal * ((newAccount.interest / 100) * 1 / 
12)) + newAccount.accountBal;
            System.out.println ("Account balance with annual interest rate applied: " + "$" + 
Double.parseDouble (df.format (newAccount.accountBal)));
            //accounts.add (newAccount);
        } else {
            System.out.println ("Customer details mismatch. Please check customer names for 
spelling errors or case sensitivity.");
        }
    }
    return newAccount;
}
```

- A computeInterest() method that accepts a BankAccount argument. Within the method, prompt the user for the number of years the account will earn interest. The method displays the ending balance of the account at each year for the number of years entered based on the interest rate attached to the BankAccount.

```
public static void computeInterest() {
    double forecastBal, accBal;
    int predictNo;
    String[] months = new String[]{"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec"};
    System.out.println ("*************************Forecast account
balance*************************");
    for (int i = 0; i < accounts.size (); i++) {
        accBal = accounts.get (i).accountBal;
        int count = 0;
        int yearCount = 2018;
        System.out.println ("Enter monthly automatic deposit amount for bank account " + (i + 1) +
": ");
        accounts.get (i).autoDeposit = sc.nextDouble ();
        System.out.println ("Enter monthly automatic withdrawal amount for bank account " + (i + 1)
+ ": ");
        accounts.get (i).autoWithdraw = sc.nextDouble ();
        System.out.println ("Enter the no. of years to predict bank account balance with interest
applied: ");
        predictNo = sc.nextInt ();
        forecastBal = (accounts.get (i).accountBal + accounts.get (i).autoDeposit) - accounts.get
(i).autoWithdraw;
        double add = (accounts.get (i).autoDeposit) - (accounts.get (i).autoWithdraw);
        System.out.println ("****************************************ANNUAL FORECAST OF ACCOUNT" + "
" + (i + 1) + " BALANCE**********" +
                "*********************************");
        System.out.println
("____________________________________________________________________________________________
__________________");
        System.out.println ("         MONTH         " + "       YEAR       " + "      MONTHLY
STARTING BALANCE     " + "      MONTHLY ENDING BALANCE");
        System.out.println
("____________________________________________________________________________________________
__________________");
        while (count < predictNo) {
            for (String month : months) {
                System.out.print ("          " + month + "          " + "       " + yearCount + "
" + "               " +
                        Double.parseDouble (df.format (accBal)));
                forecastBal = forecastBal + add;
                accBal = forecastBal;
                System.out.println ("                       " + Double.parseDouble (df.format
(forecastBal)));
            }
            yearCount++;
            count++;
        }
        System.out.println
("____________________________________________________________________________________________
__________________\n");
    }
}
```

- A displayAccount() method that displays the details of any BankAccount object passed to it.

```
public static void displayAccount() {
    int count = 0;
    System.out.println ("***************Display bank account details.*****************");
    System.out.println ("____________________________________________________________\n");
    System.out.println ("***Bank details are displayed only if the bank account number matches with
the stored customer account details.***");
    System.out.println ("Enter customer's first name: ");
    String name1 = sc.next ();
    System.out.println ("Enter customer's last name: ");
    String name2 = sc.next ();
    for (int i = 0; i < accounts.size (); i++) {
        for (i = 0; i < customerAccount.size (); i++) {
            if ((customerAccount.get (i).fName.equalsIgnoreCase (name1)) && (customerAccount.get
(i).lName.equalsIgnoreCase (name2))) {
                System.out.println ("Account holder's full name: " +
                        customerAccount.get (i).fName + " "
                        + customerAccount.get (i).lName);
                System.out.println ("No. of accounts for customer: " + accounts.size ());
                for (i = 0; i < accounts.size (); i++) {
                    count = count + 1;
                    System.out.println ("Account " + count + " interest rate: " + accounts.get
(i).interest);
                    System.out.println ("Account " + count + " Balance with interest rate applied:
" + "$ " +
                            Double.parseDouble (df.format (accounts.get (i).accountBal)));
                }
            } else {
                System.out.println ("Customer details mismatch. Try again.");
            }
        }
    }
}
```

- A dataPersistency() method that provides writing to and reading from files to store a specified customer's bank accounts details should be added.

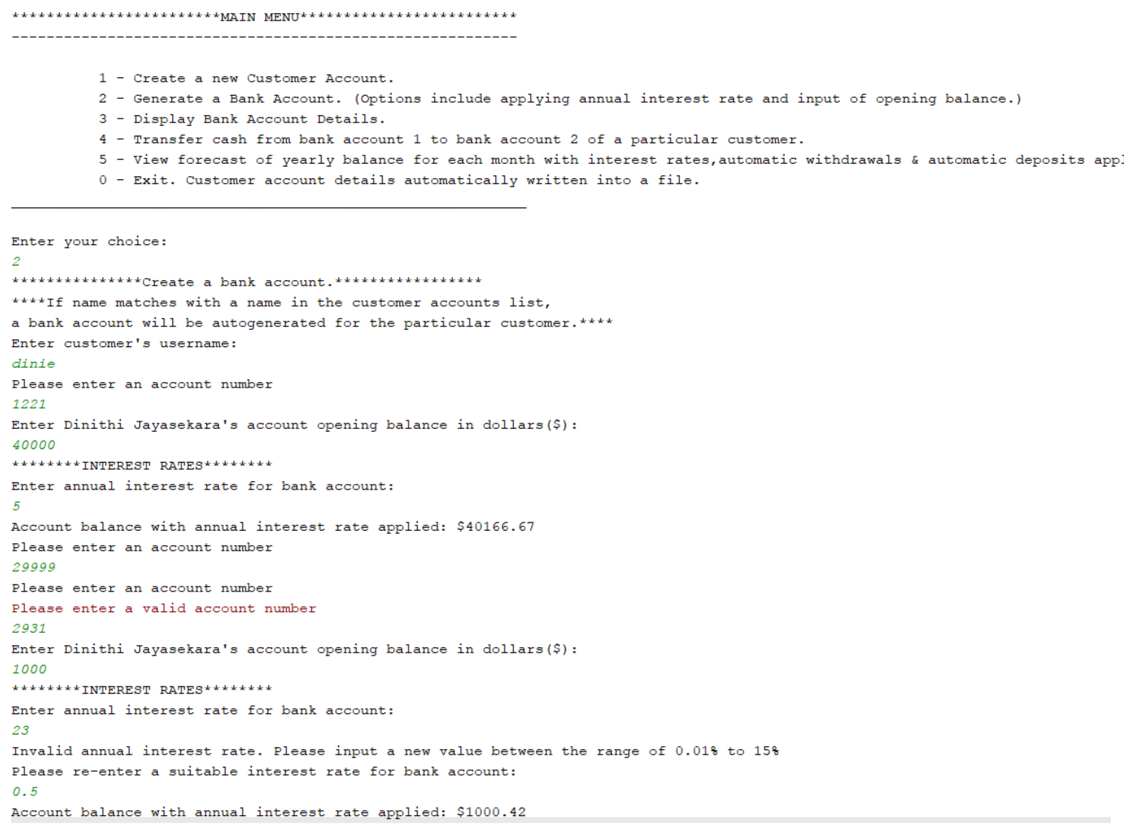
```
public static void dataPersistency() throws IOException {
    NumberFormat dformat = new DecimalFormat ("#0.00");
    String fileName = "temp.txt";
    try {
        FileWriter fileWriter = new FileWriter ("Customer_Details.txt");
        BufferedWriter bufferedWriter = new BufferedWriter (fileWriter);
        for (int i = 0; i < customerAccount.size (); i++) {
            bufferedWriter.write ("******************************CUSTOMER " + (i + 1) + "
DETAILS*************************************");
            bufferedWriter.newLine ();
            bufferedWriter.write ("Name: ");
            bufferedWriter.write (customerAccount.get (i).fName + " " + customerAccount.get
(i).lName);
            bufferedWriter.newLine ();
            bufferedWriter.write ("Username: " + customerAccount.get (i).username);
            bufferedWriter.newLine ();
            bufferedWriter.write ("Password: " + customerAccount.get (i).password);
            bufferedWriter.newLine ();
            for (i = 0; i < accounts.size (); i++) {
                bufferedWriter.write ("Total no. of bank accounts: " + accounts.size ());
                bufferedWriter.newLine ();
                for (i = 0; i < accounts.size (); i++) {
                    bufferedWriter.write ("Bank account " + (i + 1) + " interest rate: " +
accounts.get (i).interest + "%");
                    bufferedWriter.newLine ();
                    bufferedWriter.write ("Bank account " + (i + 1) + " automatic deposit amount:
" + "$" + accounts.get (i).autoDeposit);
                    bufferedWriter.newLine ();
                    bufferedWriter.write ("Bank account " + (i + 1) + " automatic withdrawal
amount: " + "$" + accounts.get (i).autoWithdraw);
                    bufferedWriter.newLine ();
                    bufferedWriter.write ("Bank account " + (i + 1) + " balance with interest rate
applied: " + "$" + dformat.format (accounts.get (i).accountBal) + "\n");
                    bufferedWriter.newLine ();
                }
            }
            bufferedWriter.close ();
        }
    }
    catch (IOException ex) {
        System.out.println ("Error writing to file.");
    }
}
```

- A main() method that declares a BankAccount object and continues to get BankAccount values from the user until the user enters a BankAccount with value 0 for the account number.

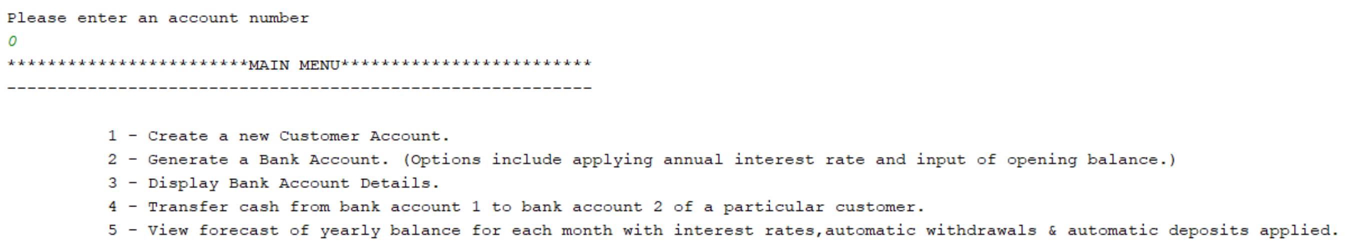
```java
public static void main(String[] args) throws IOException {
    System.out.println ("***************Welcome to Interbanking Pty.***************");
    System.out.println ("******************Account Management Unit*****************");
    System.out.println ("*************_______Login______**************\n");
    System.out.println ("Enter Username: ");
    String empName = sc.next ();
    System.out.println ("Enter Password: ");
    String empPassword = sc.next ();

    if (!Login (empName, empPassword)) {
        System.out.println ("Please enter a valid username or password");
        main (null);
    } else
        Options ();


    char choice = '\0';

    do {
        choice = sc.next ().charAt (0);
        switch (choice) {
            case '1':
                one ();
                Options ();
                break;


            case '2':
                System.out.println ("***************Create a bank account.*****************");
                System.out.println ("****If name matches with a name in the customer accounts list,
" +
                        "\na bank account will be autogenerated for the particular customer.****");
                System.out.println ("Enter customer's username: ");
                String user = sc.next ();

                while (true) {
                    System.out.println ("Please enter an account number");
                    String acc;
                    acc = sc.next ();
                    if (acc.isEmpty () || !acc.matches ("[0-9]+") || Double.parseDouble (acc) >
10000) {
                        System.err.println ("Please enter a valid account number");
                        continue;
                    } else if (Integer.parseInt (acc) == 0) {
                        break;

                    } else if (Integer.parseInt (acc) < 1000 || Integer.parseInt (acc) > 9999) {
                        System.err.println ("Account number must be between 1000 - 9999");
                        continue;
                    } else {
                        BankAccount newAcc = enterAccountData (user, Integer.parseInt (acc));
                        accounts.add (newAcc);
                    }
                }

                Options ();
                break;


            case '3':
                displayAccount ();
                Options ();
                break;

            case '4':
                four ();
                Options ();
                break;

            case '5':
                computeInterest ();
                Options ();
                break;
            case '6':
                dataPersistency ();
                Options ();
                break;
            case '0':
                dataPersistency ();
                System.out.println ("Thank you for your valuable service.\nAll customer details
have been written into to a file for further reference.");
                System.exit (0);
                break;
            default:
                System.out.println ("Enter your choice: ");
        }
    } while (choice != '0');

}
```

4. SCREENSHOTS

1. Create bank accounts in a loop until zero is entered as account number along with validations applied in previous stages.

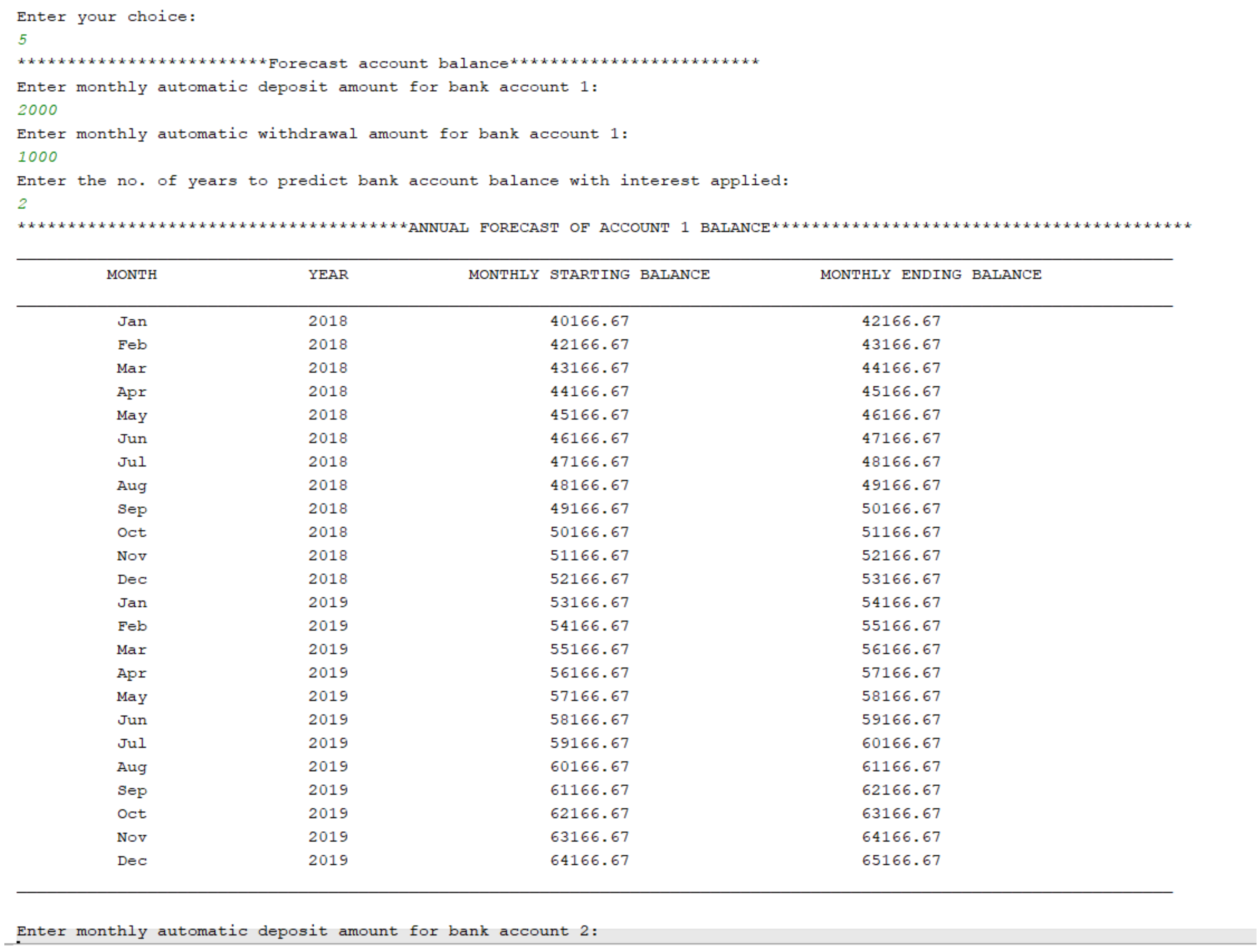


```
***********************MAIN MENU************************
---------------------------------------------------------

          1 - Create a new Customer Account.
          2 - Generate a Bank Account. (Options include applying annual interest rate and input of opening balance.)
          3 - Display Bank Account Details.
          4 - Transfer cash from bank account 1 to bank account 2 of a particular customer.
          5 - View forecast of yearly balance for each month with interest rates,automatic withdrawals & automatic deposits applied.
          0 - Exit. Customer account details automatically written into a file.
___________________________________________________________

Enter your choice:
2
***************Create a bank account.****************
****If name matches with a name in the customer accounts list,
a bank account will be autogenerated for the particular customer.****
Enter customer's username:
dinie
Please enter an account number
1221
Enter Dinithi Jayasekara's account opening balance in dollars($):
40000
********INTEREST RATES********
Enter annual interest rate for bank account:
5
Account balance with annual interest rate applied: $40166.67
Please enter an account number
29999
Please enter an account number
Please enter a valid account number
2931
Enter Dinithi Jayasekara's account opening balance in dollars($):
1000
********INTEREST RATES********
Enter annual interest rate for bank account:
23
Invalid annual interest rate. Please input a new value between the range of 0.01% to 15%
Please re-enter a suitable interest rate for bank account:
0.5
Account balance with annual interest rate applied: $1000.42
```



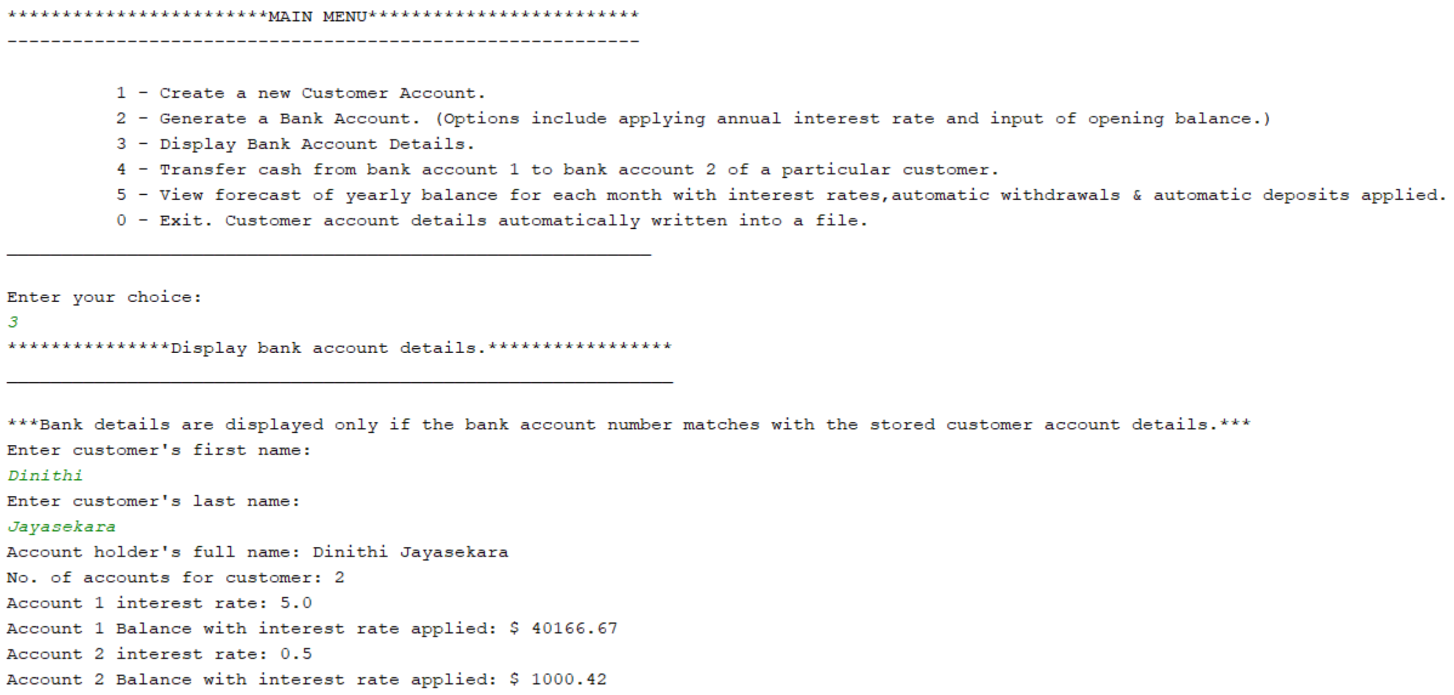
```
Please enter an account number
0
***********************MAIN MENU************************
---------------------------------------------------------

          1 - Create a new Customer Account.
          2 - Generate a Bank Account. (Options include applying annual interest rate and input of opening balance.)
          3 - Display Bank Account Details.
          4 - Transfer cash from bank account 1 to bank account 2 of a particular customer.
          5 - View forecast of yearly balance for each month with interest rates,automatic withdrawals & automatic deposits applied.
```

2. Run computeInterest() method where no. of years to display future ending balances with interest is prompted and displayed for bank accounts.



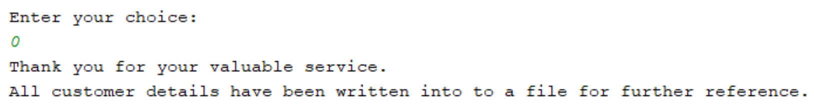
```
Enter your choice:
5
************************Forecast account balance************************
Enter monthly automatic deposit amount for bank account 1:
2000
Enter monthly automatic withdrawal amount for bank account 1:
1000
Enter the no. of years to predict bank account balance with interest applied:
2
**************************************ANNUAL FORECAST OF ACCOUNT 1 BALANCE*****************************************
_________________________________________________________________________________________________________________

        MONTH              YEAR            MONTHLY STARTING BALANCE            MONTHLY ENDING BALANCE
_________________________________________________________________________________________________________________

         Jan               2018                    40166.67                        42166.67
         Feb               2018                    42166.67                        43166.67
         Mar               2018                    43166.67                        44166.67
         Apr               2018                    44166.67                        45166.67
         May               2018                    45166.67                        46166.67
         Jun               2018                    46166.67                        47166.67
         Jul               2018                    47166.67                        48166.67
         Aug               2018                    48166.67                        49166.67
         Sep               2018                    49166.67                        50166.67
         Oct               2018                    50166.67                        51166.67
         Nov               2018                    51166.67                        52166.67
         Dec               2018                    52166.67                        53166.67
         Jan               2019                    53166.67                        54166.67
         Feb               2019                    54166.67                        55166.67
         Mar               2019                    55166.67                        56166.67
         Apr               2019                    56166.67                        57166.67
         May               2019                    57166.67                        58166.67
         Jun               2019                    58166.67                        59166.67
         Jul               2019                    59166.67                        60166.67
         Aug               2019                    60166.67                        61166.67
         Sep               2019                    61166.67                        62166.67
         Oct               2019                    62166.67                        63166.67
         Nov               2019                    63166.67                        64166.67
         Dec               2019                    64166.67                        65166.67
_________________________________________________________________________________________________________________

Enter monthly automatic deposit amount for bank account 2:
```

3. Run displayAccount() method to display details of all bank account objects.

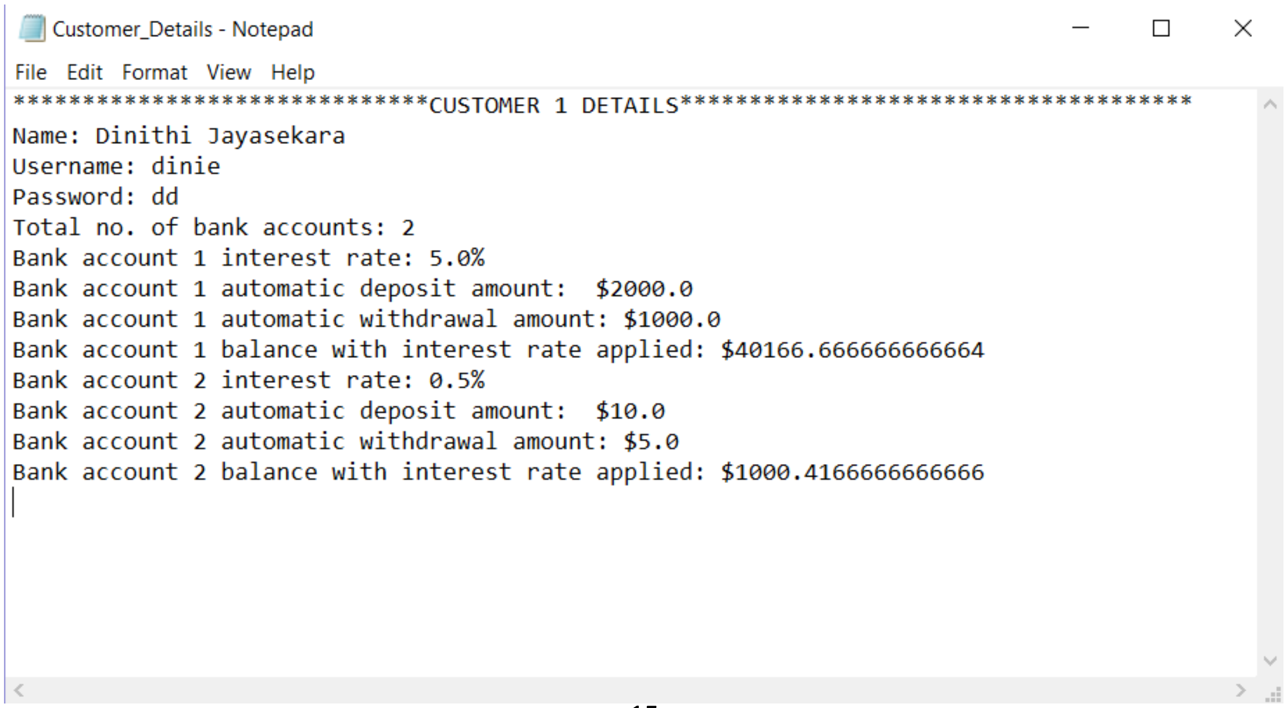


```
***********************MAIN MENU************************
--------------------------------------------------------


         1 - Create a new Customer Account.
         2 - Generate a Bank Account. (Options include applying annual interest rate and input of opening balance.)
         3 - Display Bank Account Details.
         4 - Transfer cash from bank account 1 to bank account 2 of a particular customer.
         5 - View forecast of yearly balance for each month with interest rates,automatic withdrawals & automatic deposits applied.
         0 - Exit. Customer account details automatically written into a file.
__________________________________________________________

Enter your choice:
3
***************Display bank account details.****************
____________________________________________________________

***Bank details are displayed only if the bank account number matches with the stored customer account details.***
Enter customer's first name:
Dinithi
Enter customer's last name:
Jayasekara
Account holder's full name: Dinithi Jayasekara
No. of accounts for customer: 2
Account 1 interest rate: 5.0
Account 1 Balance with interest rate applied: $ 40166.67
Account 2 interest rate: 0.5
Account 2 Balance with interest rate applied: $ 1000.42
```

4. Run dataPersistency() method to write into file.



```
Enter your choice:
0
Thank you for your valuable service.
All customer details have been written into to a file for further reference.
```



```
******************************CUSTOMER 1 DETAILS*************************************
Name: Dinithi Jayasekara
Username: dinie
Password: dd
Total no. of bank accounts: 2
Bank account 1 interest rate: 5.0%
Bank account 1 automatic deposit amount:  $2000.0
Bank account 1 automatic withdrawal amount: $1000.0
Bank account 1 balance with interest rate applied: $40166.666666666664
Bank account 2 interest rate: 0.5%
Bank account 2 automatic deposit amount:  $10.0
Bank account 2 automatic withdrawal amount: $5.0
Bank account 2 balance with interest rate applied: $1000.4166666666666
```

5. CONCLUSION

All required functions were implemented in stage 3. In this stage, the method names were changed according to the required specification and the new function of creating many bank accounts until zero is entered was implemented in the main method.