LAMBTON COLLEGE IN TORONTO

Mobile Application Design and Development

PRIMITIVE BANKING SYSTEM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROJECT DOCUMENTATION

COURSE – MAD 3463

PROGRAMMING JAVA

PROJECT CONTRIBUTORS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ASWINI SASI KANTH KANDURI -- C0880827

MANDIP KAUR -- C0880781

DHRUVRAJSINH RATHOD -- C0868147

PROFESSOR: PEDRAM FAHIGI

**INDEX**

1. Introduction …………………………………………………………………………………………………………. 03

2. Code Description ………………………………………………………………………………………………..... 03

3. Exceptions usage ………………………………………………………………………………………………….. 05

3. Code files …………………………………………………………………………………….……………………….. 05

4. Git-hub Link ………………………………………………………………………………………………………….. 22

5. Screenshots ………………………………………………………………………………………………………..… 22

6. References …………………………………………………………………………………………………………..… 29

**Introduction:**

This is a simple primitive banking software program which simulates banking system operations written in Java language. The app has a simple menu that allows the user to perform the basic banking operation like adding customers, checking data of existing customers, deposit and withdraw of amount.

Creating a new account simply requires the user to enter their name and initial deposit amount. The account number is automatically generated.

Viewing the account balance displays the current balance of the account.

Depositing and withdrawing money require the user to enter the amount they wish to deposit/withdraw. The app then updates the account balance accordingly.

The exit option just terminates the app.

**Code description:**

*Step 1:* User will be prompted with three options while entering the bank management system which will be executed by calling start() function in Start.java.

* In this function, object of Transaction will be created and following options will be shown.

What do you want to do?

1: Open New account

2: Do you have existing account?

q: Exit

* While opening account, user needs to enter the full name and opening balance. Transaction.java will open new account by storing the user information in “MyFile.txt”.
* If user choses to have existing account, existingUserData() will be called and prompt options.

Q: Choose any one of the operations to perform

1: Transaction

2: View Existing Account Information

3: Pay Utility Bills

q: Exit

*Step 2:* Choosing “Transaction” option, user will have three options Deposit, Withdraw and Transfer. For all options, user needs to enter account number and the amount. In case of “Transfer”, additional account number will be required.

* Operation == “Deposit”: call findUpdate() search for the account number provided by the user, if found add the amount and update the closing balance in file.
* Operation == “Withdraw”: call findUpdate() search for the account number provided by the user, if found subtract the amount and update the closing balance in file.
* Operation == “Transfer”: call transferAmount(), checkIfAccountExists() method will return account balance associated with the account number else return 0 if not found. After checking existence of the account updateAccountInfo() will update the closing balance of both accounts.

*Step 3:* Choosing “View Existing Account Information” option, user needs to enter account number and call findDisplay() which will check the account number in file and display the bank account information.

*Step 4:* Choosing “Pay Utility Bills” option, user needs to enter account number and amount of the bill then call payBill() which will check the account number in file and display the closing balance along with updating the file.

*Step 5:* In case of any error or wrong input given, error() function will be called to display the error.

**EXCEPTION:**

1. IOException: It will occur whenever an input or output operation is failed or interpreted.

Format of code used:

Try {

//code to be executed

//File name and location to capture the data

}

catch (IOException e) {

*//Catch input output exception*

System.*err*.println("Some error " + e.getMessage());

}

Main() class:

public class Main {

public static void main(String[] args) {

Start objectStart = new Start();

objectStart.StartBank();

}

}

Operations() class:

import java.io.BufferedReader;

import java.io.FileReader;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Start {

int count=0;

List<String> FileRecords;

public static final String *SET\_BOLD\_TEXT* = "\033[0;1m";

public void StartBank(){

@SuppressWarnings("resource")

Scanner input = new Scanner(System.*in*);

char mainYesOrNo = 'Y';

while (mainYesOrNo =='Y'){

System.*out*.print("\t=========================================\n");

System.*out*.print(*SET\_BOLD\_TEXT* +"\t \tWELCOME TO BANK MANAGEMENT SYSTEM \n");

if(count<1){

count=count+1;

}else{

System.*out*.print("\t\t Again \n");

}

System.*out*.print("\t=========================================\n\n");

if(count<2){

count=count+1;

}else{

System.*out*.print("\t\t1 : Start Again\n\n");

System.*out*.print("\t\t2 : Exit\n\n");

System.*out*.println("\tYou Select : ");

}

*//start the operation*

start();

System.*out*.println("\n\tDo u want to run your Program Again \n\t\t\tY = yes\n\t\t\tN = No\n");

System.*out*.println("You Select : ");

mainYesOrNo =(input.next()).charAt(0);

if(Character.*isLowerCase*(mainYesOrNo )){

mainYesOrNo =Character.*toUpperCase*(mainYesOrNo );

}

}

}

private void existingUserData(){

String choice, ch, operation;

Transaction transac = new Transaction();

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

double amount;

long accountNo=0;

long transferAccountNo=0;

do {

System.*out*.print("\tQ: Choose any one of the operation to perform\n\n");

System.*out*.print("\t1 : Transaction\n\n");

System.*out*.print("\t2 : View Existing Account Information\n\n");

System.*out*.print("\t3 : Pay Utility Bills\n\n");

System.*out*.print("\tq : Exit\n\n");

System.*out*.println("Your choice: ");

choice = sc.next();

switch (choice) {

case "1":

System.*out*.print("\tQ: What do you want to do for Transaction?\n\n");

System.*out*.print("\t\ta : Deposit\n\n");

System.*out*.print("\t\tb : Withdraw\n\n");

System.*out*.print("\t\tc : Transfer\n\n");

ch = sc.next();

if (ch.equalsIgnoreCase("a"))

operation = "Deposit";

else if (ch.equalsIgnoreCase("b"))

operation = "Withdraw";

else if (ch.equalsIgnoreCase("c")) {

operation = "Transfer";

System.*out*.println("Please enter Account Number to transfer the amount:");

transferAccountNo = sc.nextLong();

transac.storeTransferAccount(transferAccountNo);

}

else {

operation = "Invalid option";

}

System.*out*.println("Please enter your Account Number:");

accountNo = sc.nextLong();

System.*out*.println("Please enter Amount:");

amount = sc.nextDouble();

transac.transaction("",accountNo, operation, amount);

break;

case "2":

System.*out*.println("Account Number:");

accountNo = sc.nextLong();

operation = "showInfo";

transac.transaction("",accountNo, operation, 0);

break;

case "3":

operation = "PayBill";

System.*out*.println("Please enter your Account Number:");

accountNo = sc.nextLong();

System.*out*.println("Please enter bill amount:");

amount = sc.nextDouble();

transac.transaction("",accountNo, operation, amount);

break;

case "q":

System.*out*.println("====== Thank you! ======");

return;

default:

*error*();

}

}while (choice != "q");

sc.close();

}

private void start() {

String choice, ch, operation;

Transaction transac = new Transaction();

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

double amount;

long accountNo=0;

long transferAccountNo=0;

do {

System.*out*.print("\t======== What do you want to do? ========\n\n");

System.*out*.print("\t========= 1 : Open New account ==========\n\n");

System.*out*.print("\t===== 2 : Do you have existing account? ====\n\n");

System.*out*.print("\t============ q : Exit ============\n\n");

System.*out*.println("Your choice: ");

choice = sc.next();

switch (choice) {

case "1":

double openingBalance;

System.*out*.println("Enter your full name :");

sc.nextLine();

String name= sc.nextLine();

System.*out*.println("Enter the opening balance :");

openingBalance = sc.nextDouble();

transac.transaction(name, accountNo, "Opening", openingBalance);

accountNo = accountNo+1;

break;

case "2":

this.existingUserData();

case "q":

System.*out*.println("====== Thank you! ======");

return;

default:

*error*();

}

} while (choice != "q");

sc.close();

}

public static void error() {

System.*out*.print("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

System.*out*.print("\t\t You Select some thing wrong\n");

System.*out*.print("\t\t OR\n");

System.*out*.print("\t\t There may be some other Problem\n");

System.*out*.print("\t\t It is better for you to try again...!\n");

System.*out*.print("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

}

}

Transaction.java

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Date;

import java.util.Scanner;

public class Transaction {

private String transactionType;

private String userName;

private double amount;

private long accountNum;

private long transferAccountNum;

private Date date;

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

public Transaction() {

}

public void transaction(String userName, long accountNum, String transactionType, double amount) {

this.userName= userName;

this.accountNum= accountNum;

this.transactionType = transactionType;

this.amount = amount;

date = new Date();

operation();

}

public void storeTransferAccount(long transferAccount){

this.transferAccountNum=transferAccount;

}

private void operation() {

if (transactionType.equalsIgnoreCase("Opening")) {

try{

FileWriter fstream = new FileWriter("MyFile.txt", true);

BufferedWriter out = new BufferedWriter(fstream);

int userId=(int) (findMaxId()+1);

out.write(Integer.*toString*(userId)+"\n");

out.write(amount+"\n");

out.write(date+"\n");

out.write(userName+"\n");

out.close();

*//Account opened*

System.*out*.println("==== Congratulations! Account has been successfully opened ====");

System.*out*.println("\t\tUser Name: "+userName);

System.*out*.println("\t\tAccount Number: "+userId);

System.*out*.println("\t\tOpening Balance: "+amount+"\n");

}catch (IOException e) {

*//Catch input output exception*

System.*err*.println("Caught IOException: " + e.getMessage());

}

}

else if (transactionType.equalsIgnoreCase("withdraw")) {

Path path = Paths.*get*("MyFile.txt");

*//Check whether files exists storing account information*

if (Files.*exists*(path)) {

findUpdate() ;

}else{

System.*out*.println("File not Found");

}

}

else if (transactionType.equalsIgnoreCase("deposit")) {

*//Check whether files exists storing account information*

Path path = Paths.*get*("MyFile.txt");

if (Files.*exists*(path)) {

findUpdate() ;

}else{

System.*out*.println("File not Found");

}

}

else if (transactionType.equalsIgnoreCase("Paybill")) {

*//Check whether files exists storing account information*

Path path = Paths.*get*("MyFile.txt");

if (Files.*exists*(path)) {

payBill() ;

}else{

System.*out*.println("File not Found");

}

}

else if (transactionType.equalsIgnoreCase("transfer")) {

*//Check whether files exists storing account information*

Path path = Paths.*get*("MyFile.txt");

if (Files.*exists*(path)) {

transferAmount() ;

}else{

System.*out*.println("File not Found");

}

}

else if (transactionType.equalsIgnoreCase("showInfo")) {

Path path = Paths.*get*("MyFile.txt");

if (Files.*exists*(path)) {

findDisplay();

}else{

System.*out*.println("File not Found");

}

}

else {

System.*out*.println("Invalid option");

return;

}

}

public double checkIfAccountExists(long accountNum) {

try{

BufferedReader reader = new BufferedReader(new FileReader("MyFile.txt"));

String line;

String accountTmpNum=Long.*toString*(accountNum);

int count=1;

while ((line = reader.readLine()) != null)

{

if(count>1){

accountTmpNum =line;

}

if((line.equals(accountTmpNum))&&(count<5)){

double temp\_amount = 0;

if (count==2){

temp\_amount = Double.*parseDouble*(line);

return temp\_amount;

}

count=count+1;

}

}

}catch (Exception e)

{

}

return 0.0;

}

private void payBill() {

try{

BufferedReader reader = new BufferedReader(new FileReader("MyFile.txt"));

FileWriter fstream = new FileWriter("TempFile.txt", true);

BufferedWriter out = new BufferedWriter(fstream);

String line;

String trmpaccountNum=Long.*toString*(accountNum);

int count=1;

while ((line = reader.readLine()) != null)

{

if(count>1){

trmpaccountNum =line;

}

if((line.equals(trmpaccountNum))&&(count<4)){

double temp\_amount = 0;

if (count==1){

out.write(accountNum+"\n");

}

else if (count==2){

*//subtract the bill amount from the balance*

temp\_amount=Double.*parseDouble*(line)-amount;

if(temp\_amount<0){

System.*out*.println("You do not have sufficient balance \n");

out.write(Double.*toString*(amount)+"\n");

}else{

System.*out*.println("\n\t==== Successfully paid bill ====");

System.*out*.println("\t"+Double.*toString*(temp\_amount)+" is your closing balance\n");

out.write(Double.*toString*(temp\_amount)+"\n");

}

}

else if(count==3){

out.write(date+"\n");

}

count=count+1;

}else{

out.write(line+"\n");

}

}

out.close();

reader.close();

File f1 = new File("MyFile.txt");

f1.delete();

File f2 = new File("TempFile.txt");

boolean b = f2.renameTo(f1);

if(b){

}else{

System.*out*.println("Updated has Error");

}

}

catch (Exception e)

{

System.*err*.format("Exception occurred trying to read '%s'.", "MyFile.txt");

e.printStackTrace();

}

}

public void transferAmount(){

try{

double transferAccountBalance = this.checkIfAccountExists(transferAccountNum);

double temp\_amount = this.checkIfAccountExists(accountNum);

if(this.checkIfAccountExists(transferAccountNum)==0.0){

System.*out*.println("\nError: Transfer account number provided by you does not exist\n");

}else if(this.checkIfAccountExists(accountNum)==0.0){

System.*out*.println("\nError: Account number provided by you does not exist \n");

}else{

if(temp\_amount>amount){

*//update both account balances*

double transAccountBal = transferAccountBalance+amount;

double userAccountBal = temp\_amount-amount;

this.updateAccountInfo(accountNum, amount,"withdraw" );

this.updateAccountInfo(transferAccountNum, amount,"deposit" );

System.*out*.println("\n======== Successful operation ========");

System.*out*.println("Closing balance of Account number: "+accountNum+" is "+userAccountBal);

System.*out*.println("Closing balance of Account number: "+transferAccountNum+" is "+transAccountBal+"\n");

}else if(accountNum==transferAccountNum){

System.*out*.println("\nError: You cannot transfer amount in same account \n");

}else{

System.*out*.println("\nError: Your account number does not have sufficient balance \n");

}

}

}

catch (Exception e)

{

System.*err*.format("Exception occurred trying to read '%s'.", "MyFile.txt");

e.printStackTrace();

}

}

private int findMaxId() {

try{

BufferedReader reader = new BufferedReader(new FileReader("MyFile.txt"));

int count=0;

while ((reader.readLine()) != null)

{

count=count+1;

}

reader.close();

*// Logic for finding maximum Id*

return count/3;

}

catch (Exception e)

{

System.*err*.format("Exception occurred trying to read '%s'.", "MyFile.txt");

e.printStackTrace();

}

return 0;

}

private void findDisplay() {

try{

BufferedReader reader = new BufferedReader(new FileReader("MyFile.txt"));

String line;

String trmpaccountNum=Long.*toString*(accountNum);

int count=1;

System.*out*.println("\n======== Your Account Information ========\n");

while ((line = reader.readLine()) != null)

{

if(count>1){

trmpaccountNum =line;

}

if((line.equals(trmpaccountNum))&&(count<5)){

switch (count){

case 1: System.*out*.println("Account Number: "+line);

break;

case 2: System.*out*.println("Balance: "+line);

break;

case 3: System.*out*.println("Date of opening: "+line);

break;

case 4: System.*out*.println("Name of the account holder: "+line+"\n");

break;

}

count=count+1;

}

}

reader.close();

}

catch (Exception e)

{

System.*err*.format("Exception occurred trying to read '%s'.", "MyFile.txt");

e.printStackTrace();

}

}

private void updateAccountInfo(long accountNum, double amount, String type){

try{

BufferedReader reader = new BufferedReader(new FileReader("MyFile.txt"));

FileWriter fstream = new FileWriter("TempFile.txt", true);

BufferedWriter out = new BufferedWriter(fstream);

String line;

String trmpaccountNum=Long.*toString*(accountNum);

int count=1;

while ((line = reader.readLine()) != null)

{

if(count>1){

trmpaccountNum =line;

}

if((line.equals(trmpaccountNum))&&(count<4)){

double temp\_amount = 0;

if (count==1){

out.write(accountNum+"\n");

}

else if (count==2){

if(type.equalsIgnoreCase("withdraw")){

*//subtracts the withdrawn amount*

temp\_amount=Double.*parseDouble*(line)-amount;

}else if(type.equalsIgnoreCase("deposit")){

*//add the deposit amount*

temp\_amount=amount+Double.*parseDouble*(line);

}

if(temp\_amount<0){

System.*out*.println("You do not have sufficient balance \n");

out.write(Double.*toString*(amount)+"\n");

}else{

out.write(Double.*toString*(temp\_amount)+"\n");

}

}

else if(count==3){

out.write(date+"\n");

}

count=count+1;

}else{

out.write(line+"\n");

}

}

out.close();

reader.close();

File f1 = new File("MyFile.txt");

f1.delete();

File f2 = new File("TempFile.txt");

boolean b = f2.renameTo(f1);

if(b){

}else{

System.*out*.println("Updated has Error");

}

}

catch (Exception e)

{

System.*err*.format("Exception occurred trying to read '%s'.", "MyFile.txt");

e.printStackTrace();

}

}

private void findUpdate() {

try{

BufferedReader reader = new BufferedReader(new FileReader("MyFile.txt"));

FileWriter fstream = new FileWriter("TempFile.txt", true);

BufferedWriter out = new BufferedWriter(fstream);

String line;

String trmpaccountNum=Long.*toString*(accountNum);

int count=1;

while ((line = reader.readLine()) != null)

{

if(count>1){

trmpaccountNum =line;

}

if((line.equals(trmpaccountNum))&&(count<4)){

double temp\_amount = 0;

if (count==1){

out.write(accountNum+"\n");

}

else if (count==2){

if(transactionType.equalsIgnoreCase("withdraw")){

*//subtracts the withdrawn amount*

temp\_amount=Double.*parseDouble*(line)-amount;

}else if(transactionType.equalsIgnoreCase("deposit")){

*//add the deposit amount*

temp\_amount=amount+Double.*parseDouble*(line);

}

if(temp\_amount<0){

System.*out*.println("You do not have sufficient balance \n");

out.write(Double.*toString*(amount)+"\n");

}else{

System.*out*.println("==== Successful operation ====");

System.*out*.println(Double.*toString*(temp\_amount)+" is your closing balance\n");

out.write(Double.*toString*(temp\_amount)+"\n");

}

}

else if(count==3){

out.write(date+"\n");

}

count=count+1;

}else{

out.write(line+"\n");

}

}

out.close();

reader.close();

File f1 = new File("MyFile.txt");

f1.delete();

File f2 = new File("TempFile.txt");

boolean b = f2.renameTo(f1);

if(b){

}else{

System.*out*.println("Updated has Error");

}

}

catch (Exception e)

{

System.*err*.format("Exception occurred trying to read '%s'.", "MyFile.txt");

e.printStackTrace();

}

}

}

**GitLink:** [**https://github.com/aswinsasikanth/JavaFinalProject.git**](https://github.com/aswinsasikanth/JavaFinalProject.git)

**SCREENSHOTS:**

Screenshot 1: Initial presentation of the available options to the user.

Text

Description automatically generated

Screenshot 2: Selecting option 1 “Open new account”

Table

Description automatically generated with low confidence

Text

Description automatically generated

Screenshot 3: Selecting option 2 “Do you have existing account?”

Text

Description automatically generated with medium confidence

Screenshot 4: Selecting option Transaction from the above options

Graphical user interface, text

Description automatically generated

Screenshot 5: Selecting option Deposit

Text

Description automatically generated

Screenshot 6: Selecting option Withdraw

Graphical user interface, text

Description automatically generated

Screenshot 7: Selecting option Transfer

Graphical user interface, text, application

Description automatically generated

Screenshot 8: Selecting option “View Existing Account Information”

Graphical user interface, text, application, email

Description automatically generated

Screenshot 9: Selecting option “Pay utility bills”

Text

Description automatically generated

Screenshot 10: Selecting option “Exit”

Graphical user interface, text, application

Description automatically generated

Screenshot 11: File storing user bank details

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Screenshot 12: Error in case of wrong input

Graphical user interface, text

Description automatically generated with medium confidence

**Class Diagram:**

Diagram

Description automatically generated

**Reference:**

[**https://docs.oracle.com/javase/tutorial/essential/io/**](https://docs.oracle.com/javase/tutorial/essential/io/)

[**https://www.tutorialspoint.com/java/java\_files\_io.htm**](https://www.tutorialspoint.com/java/java_files_io.htm)

[**https://www.youtube.com/watch?v=wRC01C0Q5o0**](https://www.youtube.com/watch?v=wRC01C0Q5o0)

[**https://www.youtube.com/watch?v=-HZSyjuuOsc&t=796s**](https://www.youtube.com/watch?v=-HZSyjuuOsc&t=796s)