Comsats University Islamabad, Lahore Campus

Object Oriented Programming

Assignment 02

Submitted by

- Aamir Mubeen (FA23-BSE-032)
- Shoaib Shahid (FA23-BSE-137)
- Mateen Haider(FA22-BSE-209)

Mam Muntaha Iqbal

Submission Date

April 20 , 2024

Account Class:

```java
import java.io.IOException;
import java.util.ArrayList;


public  class Account {
    private String accountNumber;
    private double balance;
    private String accountType;
    private final Customer accountHolder;
    private String atmPin;

    public Account(String accountNumber, double balance, String
accountType, Customer accountHolder, String atmPin) {
        this.accountNumber = accountNumber;
        this.balance = balance;
        this.accountType = accountType;
        this.accountHolder = accountHolder;

        this.atmPin = atmPin;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public String getAccountType() {
        return accountType;
    }

    public void setAccountType(String accountType) {
        this.accountType = accountType;
    }

    public Customer getAccountHolder() {
        return accountHolder;
    }

    public void setAtmPin(String atmPin) {
        this.atmPin = atmPin;
    }

    public String getAtmPin() {
        return this.atmPin;
    }



}
```

Administrator Class:

```
public class Administrator {
    private String adminId;
    private String adminName;
    private String adminEmail;
    private String adminPass;

    public Administrator(String adminId, String adminName, String
adminEmail, String adminPass) {
        this.adminId = adminId;
```

```java
        this.adminName = adminName;
        this.adminEmail = adminEmail;
        this.adminPass = adminPass;
    }

    public String getAdminName() {
        return adminName;
    }

    public void setAdminName(String adminName) {
        this.adminName = adminName;
    }

    public String getAdminEmail() {
        return adminEmail;
    }

    public void setAdminEmail(String adminEmail) {
        this.adminEmail = adminEmail;
    }

    public String getAdminPass() {
        return adminPass;
    }

    public void setAdminPass(String adminPass) {
        this.adminPass = adminPass;
    }

    public String getAdminId() {
        return adminId;
    }

    public void setAdminId(String adminId) {
        this.adminId = adminId;
    }
    public void approveAccount(Account account) {
    }
    public void blockAccount(Account account) {
    }

    public void generateReports() {
    }
}
```

ATM class:

```java
public class ATM {
    private String atmId;
    private String location;
    private double cashAvailable;

    public ATM(String atmId, String location, double initialCash)
{
        this.atmId = atmId;
        this.location = location;
        this.cashAvailable = initialCash;
    }

    public String getAtmId() {
        return atmId;
    }

    public void setAtmId(String atmId) {
        this.atmId = atmId;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }
    public double getCashAvailable() {
        return cashAvailable;
    }

    public void setCashAvailable(double cashAvailable) {
        this.cashAvailable = cashAvailable;
    }

    public String withdrawCashSavingAcc(double amount,
SavingAccount savingAcc) {

        if (cashAvailable >= amount && (savingAcc.getBalance() >=
amount)) {
            cashAvailable -= amount;
            savingAcc.setBalance(savingAcc.getBalance()-amount);
            return "Transaction successfull! Please take your
Rs." + amount;
        }
        return  "Transaction failed due to insufficient cash in
ATM machine";
    }
    public String withdrawCashCurrentAcc(double amount,
CurrentAccount currentAcc){
    // for saving account
        if(cashAvailable >= amount){
            if(currentAcc.getBalance() >= amount){
                cashAvailable -= amount;
                currentAcc.setBalance(currentAcc.getBalance()-
amount);
                return "Transaction successfull! Please take your
Rs." + amount;
            }
            else{
                if(currentAcc.getRemainingOverdraftLimit() >=
amount){
                    cashAvailable -= amount;

currentAcc.setRemainingOverdraftLimit(currentAcc.getRemainingOver
draftLimit()-amount);
```

```
                    return "Transaction successfull! Please take
your Rs." + amount;
                }
                return "Transaction failed due to insufficient
balance in your account";
            }
        }
        return  "Transaction failed due to insufficient cash in
ATM machine";
    }

    public void depositCash(double amount, Account account) {
        cashAvailable += amount;
        double currentBalance = account.getBalance();
        double newBalance = currentBalance + amount;
        account.setBalance(newBalance);
    }
}
```

Bank Class:

```
import java.util.ArrayList;
import java.util.List;

public class Bank {
    private String bankName;
    private String bankAddress;
    private List<Customer> customers;
    private List<Account> accounts;

    public Bank(String bankName, String bankAddress) {
        this.bankName = bankName;
        this.bankAddress = bankAddress;
        this.customers = new ArrayList<>();
        this.accounts = new ArrayList<>();
    }
```

```java
    public String getBankName() {
        return bankName;
    }

    public void setBankName(String bankName) {
        this.bankName = bankName;
    }

    public String getBankAddress() {
        return bankAddress;
    }

    public void setBankAddress(String bankAddress) {
        this.bankAddress = bankAddress;
    }

    public void addCustomer(Customer customer) {
        customers.add(customer);
    }

    public void removeCustomer(Customer customer) {
        customers.remove(customer);
    }

    public List<Customer> getCustomers() {
        return customers;
    }

    public void addAccount(Account account) {
        accounts.add(account);
    }

    public void removeAccount(Account account) {
        accounts.remove(account);
    }

    public List<Account> getAccounts() {
        return accounts;
    }
}
```

BillPaymentServices Class:

```
public class BillPaymentServices {

//    public  boolean payElectricityBill(String accNum, String
provider, double amount) {
////        if (account.getBalance() >= amount) {
////            account.setBalance(account.getBalance() -
amount);
////            return true;
////        }
//       return false;
//    }
//
//    public  boolean payEducationBill(String accNum, String
institution, double amount) {
////        if (account.getBalance() >= amount) {
////            account.setBalance(account.getBalance() -
amount);
////            return true;
////        }
//      return false;
//    }
//
//    public  boolean payWaterBill(String accNum, String
utilityCompany, double amount) {
////        if (account.getBalance() >= amount) {
////            account.setBalance(account.getBalance() -
amount);
////            return true;
////        }
//       return false;
//    }
//
//    public  boolean payGasBill(String accNum, String provider,
double amount) {
```

```
////        if (account.getBalance() >= amount) {
////            account.setBalance(account.getBalance() -
amount);
////            return true;
////        }
//        return false;
//    }
//    public  boolean payInternetBill(String accNum, String
provider, double amount) {
////        if (account.getBalance() >= amount) {
////            account.setBalance(account.getBalance() -
amount);
////            return true;
////        }
//      return false;
//    }
//
//    public  boolean payOtherBill(String accNum, String payee,
double amount) {
////        if (account.getBalance() >= amount) {
////            account.setBalance(account.getBalance() -
amount);
////            return true;
////        }
//      return false;
//    }
    public void makeBillPayment(String accNum, String
providerName, double amount){
            Verifications v = new Verifications();
            FundsTransferServices fts = new
FundsTransferServices();
            String accType = v.getAccType(accNum);
            if(accType.equals("current")){
                boolean[] isSuccessful =
fts.debitSenderAcc(accNum, String.valueOf(amount));
                if(isSuccessful[0] && isSuccessful[1] &&
isSuccessful[2]){
                    System.out.println("The bill has successfully
been payed");
```

```
                }
                else{
                    if(!(isSuccessful[0])){
                        System.out.println("Transaction failed: 
Account not found");
                    }
                    else if(!(isSuccessful[1])){
                        System.out.println("Transaction failed: 
Balance insufficient");
                    }
                    else{
                        System.out.println("An error occurred 
while paying the bill! Please try again");
                    }
                }
            }
            else if(accType.equals("saving")){
                boolean[] isSuccessful = 
fts.debitSavingAccount(accNum,String.valueOf(amount));
                if(isSuccessful[0] && isSuccessful[1] && 
isSuccessful[2]){
                    System.out.println("The bill has successfully 
been payed");
                }
                else{
                    if(!(isSuccessful[0])){
                        System.out.println("Transaction failed: 
Account not found");
                    }
                    else if(!(isSuccessful[1])){
                        System.out.println("Transaction failed: 
Balance insufficient");
                    }
                    else{
                        System.out.println("An error occurred 
while paying the bill! Please try again");
                    }
                }
            }
            else{
                System.out.println("An unexpected error occurred!
Please try again later");
            }
    }
}
```

CreateAccount Class:

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class CreateAccount {
    public void getCustomerCredentials(String accType) {
        Scanner inp = new Scanner(System.in);
        String accNum, fName, lName, email, pass, address,
phoneNum, atmPin;
        int custId;
        double balance;
        System.out.println("Account Number:");
        accNum = inp.nextLine();

        System.out.println("Balance: ");
        balance = inp.nextDouble();
        inp.nextLine(); // Consume newline character

        System.out.println("Customer Id: ");
        custId = inp.nextInt();
        inp.nextLine(); // Consume newline character

        System.out.println("First Name: ");
        fName = inp.nextLine();
        System.out.println("Last Name: ");
        lName = inp.nextLine();
        System.out.println("Email: ");
        email = inp.nextLine();
        System.out.println("Password: ");
        pass = inp.nextLine();
        System.out.println("Address: ");
        address = inp.nextLine();
        System.out.println("Phone Number: ");
        phoneNum = inp.nextLine();
        System.out.println("Atm Pin");
        atmPin = inp.nextLine();

        Account account;
        if (accType.equalsIgnoreCase("current")) {
            System.out.println("Overdraft Limit:");
            double overdraftLimit = inp.nextDouble();
            account = new CurrentAccount(accNum, balance,
accType, new Customer(custId, fName, lName, email, pass, address,
phoneNum), overdraftLimit, atmPin);
        } else if (accType.equalsIgnoreCase("saving")) {
            System.out.println("Interest Rate:");
            double interestRate = inp.nextDouble();
            account = new SavingAccount(accNum, balance, accType,
new Customer(custId, fName, lName, email, pass, address,
phoneNum), interestRate, atmPin);
        } else {
            System.out.println("Invalid account type.");
            return;
        }

        createAccount(account);
    }

    public void createAccount(Account account) {
        try {
            FileWriter fw = new FileWriter("Files/accounts.txt",
true);
            String accNum, custId, fName, lName, email, pass,
address, phoneNum, balance, accType, specificInfo, atmPin;
            Customer accHolder = account.getAccountHolder();
            accNum = account.getAccountNumber();
            balance = Double.toString(account.getBalance());
            accType = account.getAccountType();
```

```java
            if (account instanceof CurrentAccount) {
                specificInfo = Double.toString(((CurrentAccount)
account).getOverdraftLimit());
            } else if (account instanceof SavingAccount) {
                specificInfo = Double.toString(((SavingAccount)
account).getInterestRate());
            } else {
                specificInfo = ""; // Handle other account types
if necessary
            }
            custId = Integer.toString(accHolder.getCustomerId());
            fName = accHolder.getFirstName();
            lName = accHolder.getLastName();
            email = accHolder.getEmail();
            pass = accHolder.getPassword();
            address = accHolder.getAddress();
            phoneNum = accHolder.getPhoneNumber();
            atmPin = account.getAtmPin();

            fw.write(accNum.trim() + "," + accType.trim() + "," +
specificInfo + "," + balance.trim() + "," + custId.trim() + "," +
fName.trim() + "," + lName.trim() + "," + email.trim() + "," +
pass.trim() + "," + address.trim() + "," + phoneNum.trim() + ","
+ atmPin.trim() + "\n");
            fw.close();
            System.out.println("Your account has successfully
been created! You can now login.");
        } catch (IOException e) {
            System.out.println("There was an error creating your
account! Please try again later");
            e.printStackTrace();
        }
    }
}
```

CurrentAccount Class:

```java
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class CurrentAccount extends Account {
    private double overdraftLimit;
    private double remainingOverdraftLimit;

    public CurrentAccount(String accountId, double balance,
String accountType, Customer owner, double overdraftLimit, String
atmPin) {
        super(accountId, balance, accountType, owner, atmPin);
        this.overdraftLimit = overdraftLimit;
    }

    public double getOverdraftLimit() {
        return overdraftLimit;
    }

    public void setOverdraftLimit(double overdraftLimit) {
        this.overdraftLimit = overdraftLimit;
    }
    public double getRemainingOverdraftLimit() {
        return remainingOverdraftLimit;
    }

    public void setRemainingOverdraftLimit(double
remainingOverdraftLimit) {
        this.remainingOverdraftLimit = remainingOverdraftLimit;
    }



}
```

Customer Class:

```java
public class Customer {
    private int customerId;
    private String firstName;
    private String lastName;
    private String email;
    private String password;
    private String address;
    private String phoneNumber;

    public Customer(int customerId, String firstName, String
lastName, String email,String password, String address, String
phoneNumber) {
        this.customerId = customerId;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.address = address;
        this.phoneNumber = phoneNumber;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
}
```

FundsTransferServices Class:

```java
import java.io.*;
import java.nio.channels.FileChannel;
import java.util.Scanner;

public class FundsTransferServices {

    public void fundstransfer(String senderAccNum, String
recepientAccNum, double amount) {
        boolean[] isDebitSuccessful =
this.debitSenderAcc(senderAccNum, String.valueOf(amount));
        boolean[] isCreditSuccessful =
this.creditReceiverAcc(recepientAccNum, String.valueOf(amount));
        if (isDebitSuccessful[0] && isDebitSuccessful[1] &&
isDebitSuccessful[2] && isCreditSuccessful[0] &&
isCreditSuccessful[1]) {
            System.out.println("Transaction successful");
        } else {
            if (!isDebitSuccessful[0]) {
                System.out.println("Transaction failed: Sender
account not found");
            } else if (!isDebitSuccessful[1]) {
                System.out.println("Transaction failed:
Insufficient balance in sender account");
            } else if (!isCreditSuccessful[0]) {
                System.out.println("Transaction failed: Receiver
account not found");
            } else {
                System.out.println("Transaction failed: An error
occurred while performing transaction");
            }
        }
    }
    public boolean[] debitSenderAcc(String accNum, String amount)
{
        boolean success = false;
        boolean accountFound = false;
        boolean balanceSufficient = false;
        double amountD = Double.parseDouble(amount);
        File tempFile = new File("Files/temp.txt");
        try {
            File file = new File("Files/accounts.txt");
            Scanner sc = new Scanner(file);
            FileWriter fw = new FileWriter(tempFile);
            while (sc.hasNextLine()) {
                String line = sc.nextLine();
                StringBuilder modifiedLine = new StringBuilder();
                String[] credentials = line.split(",");
                if (credentials[0].equals(accNum)) {
                    accountFound = true;
                    if (credentials[1].equals("saving")) {
                        if (Double.parseDouble(credentials[3]) >=
amountD) {
                            double oldBalance =
Double.parseDouble(credentials[3]);
                            double newBalance = oldBalance -
amountD;
                            credentials[3] =
String.valueOf(newBalance);
                            balanceSufficient = true;
                            success = true;
                        }
                    } else { // the account type is current
                        double balance =
Double.parseDouble(credentials[3]);
                        double overdraftLimit =
Double.parseDouble(credentials[2]);
                        if (balance >= amountD) {
                            balance -= amountD;
                            credentials[3] =
String.valueOf(balance);
                            balanceSufficient = true;
                            success = true;

                        }
                        else if(balance == 0 && overdraftLimit
>=0){
                            overdraftLimit -= amountD;
                            credentials[2] =
String.valueOf(overdraftLimit);
                            balanceSufficient = true;
                            success = true;
                        }
                        else if (balance < amountD && balance > 0
&& overdraftLimit >= amountD) {
                            double remainingAmount = amountD -
balance;
                            overdraftLimit -= remainingAmount;
                            balance = 0.0;
                            credentials[2] =
String.valueOf(overdraftLimit);
                            credentials[3] =
String.valueOf(balance);
                            balanceSufficient = true;
                            success = true;
                        }
                    }
                }
                for (String val : credentials) {
                    modifiedLine.append(val).append(",");
                }
                modifiedLine.deleteCharAt(modifiedLine.length() -
1);

                line = modifiedLine.toString();
                fw.write(line + "\n");
            }
            fw.close();
            sc.close();
            if (accountFound && balanceSufficient && success) {
                try (FileChannel src = new
FileInputStream(tempFile).getChannel();
                     FileChannel dest = new
FileOutputStream(file).getChannel()) {
                    dest.transferFrom(src, 0, src.size());
```

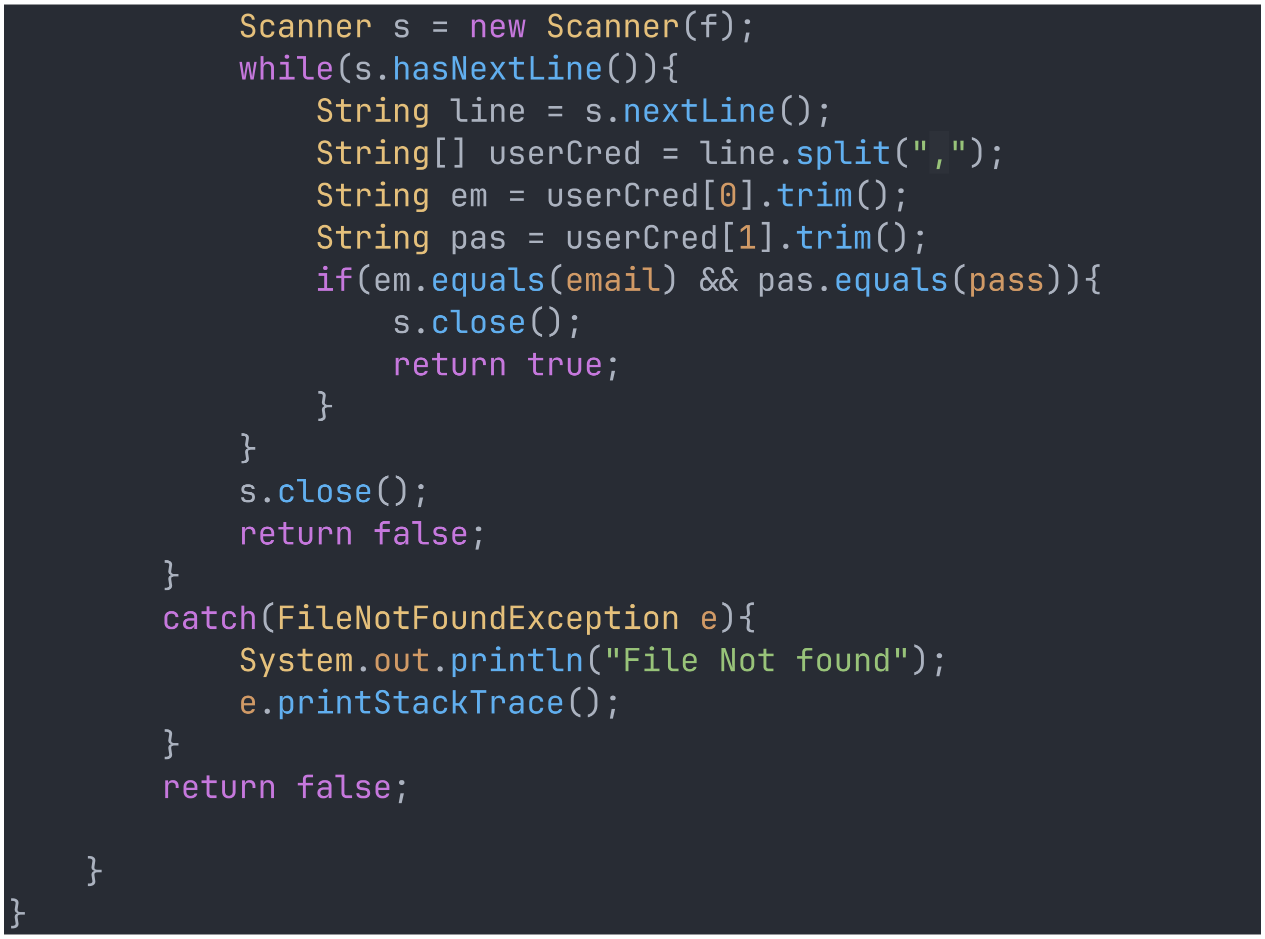
```java
                } catch (IOException e) {
                    System.out.println("An exception occurred
while performing transaction");
                }
            }
        } catch (IOException e) {
            System.out.println("An exception occurred in debiting
sender");
        } finally {
            tempFile.delete(); // Delete tempFile after
operations are completed
        }
        return new boolean[]{accountFound, balanceSufficient,
success};
    }

    public boolean[] creditReceiverAcc(String accNum, String
amount) {
        boolean success = false;
        boolean accountFound = false;
        double amountD = Double.parseDouble(amount);
        File tempFile = new File("Files/temp.txt");
        try {
            File file = new File("Files/accounts.txt");
            Scanner sc = new Scanner(file);
            FileWriter fw = new FileWriter(tempFile);
            while (sc.hasNextLine()) {
                String line = sc.nextLine();
                StringBuilder modifiedLine = new StringBuilder();
                String[] credentials = line.split(",");
                if (credentials[0].equals(accNum)) {
                    accountFound = true;
                    double oldBalance =
Double.parseDouble(credentials[3]);
                    double newBalance = oldBalance + amountD;
                    credentials[3] = String.valueOf(newBalance);
                    success = true;
                }
                for (String val : credentials) {
                    modifiedLine.append(val).append(",");
                }
                modifiedLine.deleteCharAt(modifiedLine.length() -
1);
                line = modifiedLine.toString();
                fw.write(line + "\n");
            }
            fw.close();
            sc.close();
            if (accountFound && success) {
                try (FileChannel src = new
FileInputStream(tempFile).getChannel();
                     FileChannel dest = new
FileOutputStream(file).getChannel()) {
                    dest.transferFrom(src, 0, src.size());
                } catch (IOException e) {
                    System.out.println("An exception occurred
while transferring data");
                }
            }
        } catch (IOException e) {
            System.out.println("An exception occurred in
crediting receiver");
        } finally {
            tempFile.delete(); // Delete tempFile after
operations are completed


        }
        return new boolean[]{accountFound, success};
    }
    public boolean[] debitSavingAccount(String accNum, String
amount){
        boolean success = false;
        boolean accountFound = false;
        boolean balanceSufficient = false;
        double amountD = Double.parseDouble(amount);
        File tempFile = new File("Files/temp.txt");
        try {
            File file = new File("Files/accounts.txt");
            Scanner sc = new Scanner(file);
            FileWriter fw = new FileWriter(tempFile);
            while (sc.hasNextLine()) {
                String line = sc.nextLine();
                StringBuilder modifiedLine = new StringBuilder();
                String[] credentials = line.split(",");
                if (credentials[0].equals(accNum)) {
                    accountFound = true;
                    double currentBalance =
Double.parseDouble(credentials[3]);
                    if(currentBalance >= amountD){
                        currentBalance -= amountD;
                        credentials[3] =
String.valueOf(currentBalance);
                        balanceSufficient = true;
                        success = true;
                    }
                }
                for (String val : credentials) {
                    modifiedLine.append(val).append(",");
                }
                modifiedLine.deleteCharAt(modifiedLine.length() -
1);
                line = modifiedLine.toString();
                fw.write(line + "\n");
            }
            fw.close();
            sc.close();
            if (accountFound && balanceSufficient && success) {
                try (FileChannel src = new
FileInputStream(tempFile).getChannel();
                     FileChannel dest = new
FileOutputStream(file).getChannel()) {
                    dest.transferFrom(src, 0, src.size());
                } catch (IOException e) {
                    System.out.println("An exception occurred
while performing transaction");
                }
            }
        } catch (IOException e) {
            System.out.println("An exception occurred in debiting
sender");
        } finally {
            tempFile.delete(); // Delete tempFile after
operations are completed
        }
        return new boolean[]{accountFound, balanceSufficient,
success};
    }
}
```

InterestCalculator Class:

```
public class InterestCalculator {

    public static double calculateInterest(SavingAccount
savingsAccount) {
        double balance = savingsAccount.getBalance();
        double interestRate = savingsAccount.getInterestRate();
        return balance * (interestRate / 100);
    }
}
```

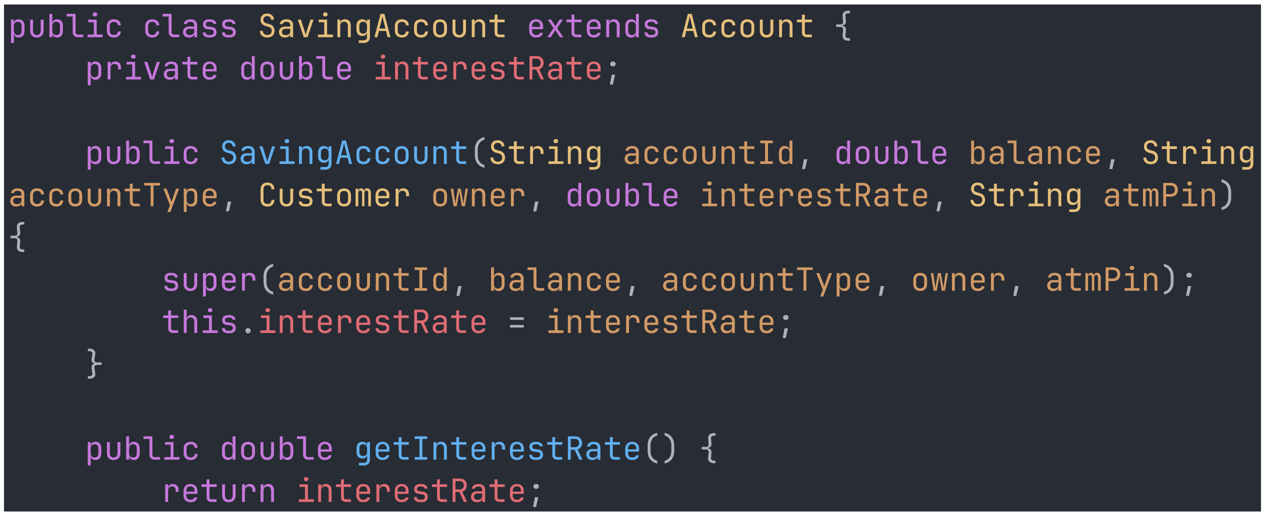
Login Class:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class Login {
    public boolean verifyLoginCredentials(String email, String
pass){
        try {
            File f = new File("Files\\customerCredentials.txt");
```



```java
            Scanner s = new Scanner(f);
            while(s.hasNextLine()){
                String line = s.nextLine();
                String[] userCred = line.split(",");
                String em = userCred[0].trim();
                String pas = userCred[1].trim();
                if(em.equals(email) && pas.equals(pass)){
                    s.close();
                    return true;
                }
            }
            s.close();
            return false;
        }
        catch(FileNotFoundException e){
            System.out.println("File Not found");
            e.printStackTrace();
        }
        return false;

    }
}
```

SavingAccount Class:



```java
public class SavingAccount extends Account {
    private double interestRate;

    public SavingAccount(String accountId, double balance, String
accountType, Customer owner, double interestRate, String atmPin)
{
        super(accountId, balance, accountType, owner, atmPin);
        this.interestRate = interestRate;
    }

    public double getInterestRate() {
        return interestRate;
    }

    public void setInterestRate(double interestRate) {
        this.interestRate = interestRate;
    }

}
```

Transaction Class:

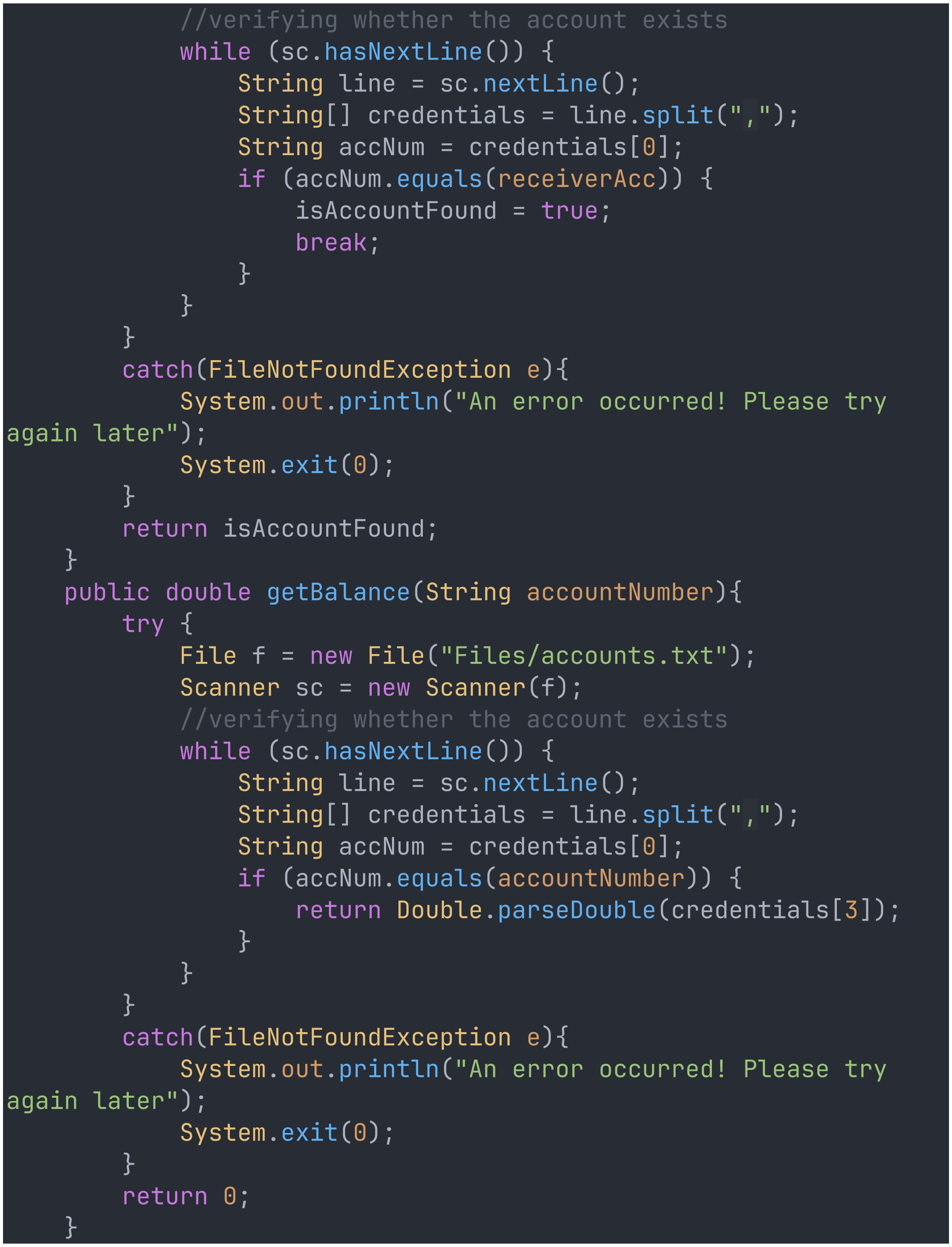
```
public class Transaction {
    private int transactionId;
    private double amount;
    private String timestamp;
    private String description;

    public Transaction(int transactionId, double amount, String
timestamp, String description) {
        this.transactionId = transactionId;
        this.amount = amount;
        this.timestamp = timestamp;
        this.description = description;
    }

    public int getTransactionId() {
        return transactionId;
    }

    public void setTransactionId(int transactionId) {
        this.transactionId = transactionId;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public String getDate() {
        return this.timestamp;
    }

    public void setDate(String timestamp) {
        this.timestamp = timestamp;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

Validations Class:

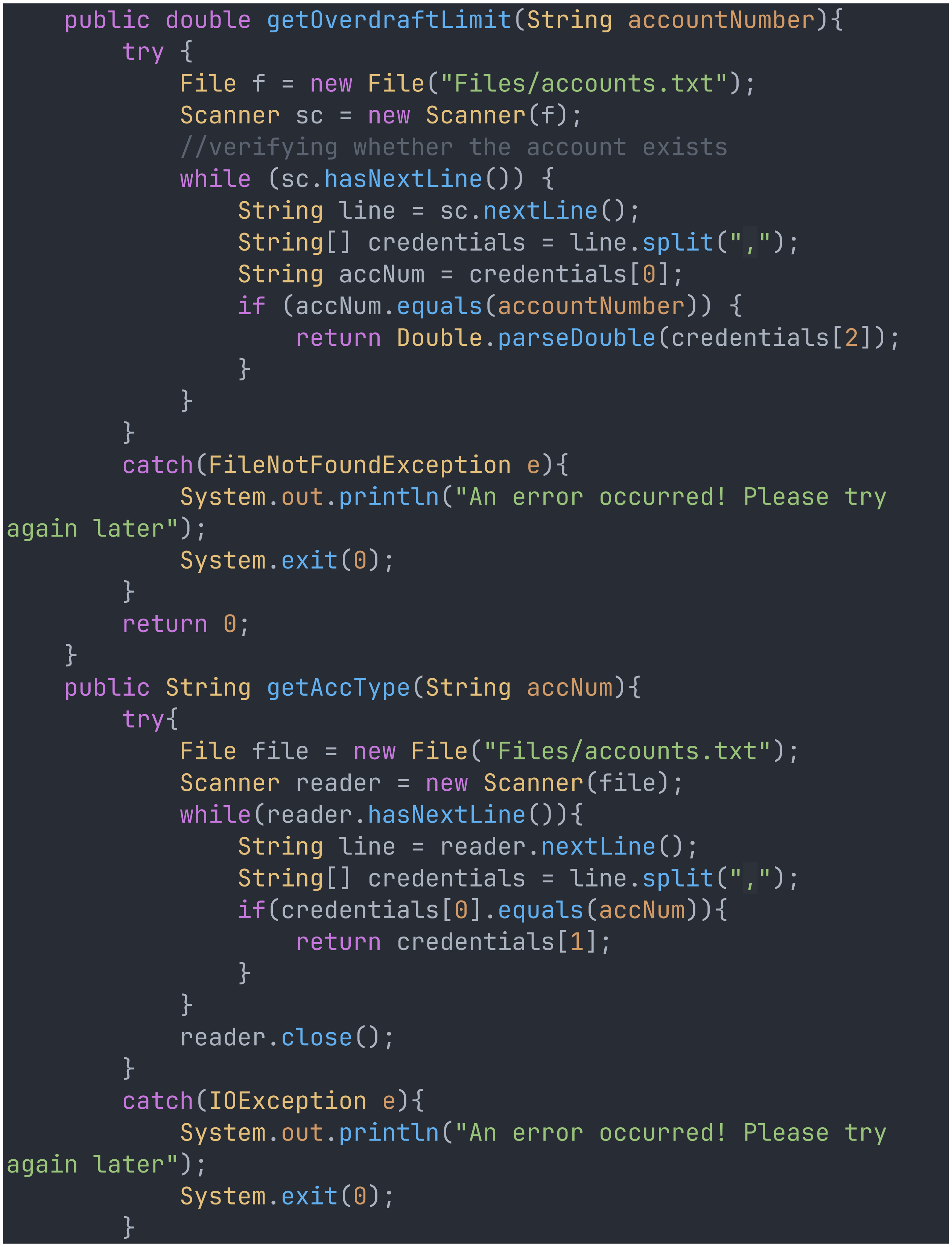
```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Validations {

    public static boolean isEmailValid(String email) {
        String emailRegex = "^[a-zA-Z0-9_+&-]+(?:\\.[a-zA-Z0-
9_+&-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";
        Pattern pattern = Pattern.compile(emailRegex);
        Matcher matcher = pattern.matcher(email);
        return matcher.matches();
    }

    public static boolean isPhoneNumberValid(String phoneNumber)
{
        String phoneRegex = "^\\d{10}$";
        Pattern pattern = Pattern.compile(phoneRegex);
        Matcher matcher = pattern.matcher(phoneNumber);
        return matcher.matches();
    }

    public static boolean isAmountValid(double amount) {
        return amount >= 0;
    }

    public static boolean isAccountNumberValid(String
accountNumber) {
        return accountNumber.matches("^\\d{10}$");
    }

    public static boolean isPasswordValid(String password) {
        return password.length() >= 8 &&
password.matches(".[!@#$%^&()].*");
    }
}
```

Verifications Class:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Scanner;
public class Verifications {

    public boolean[] verifySenderAcc(String senderAcc, String
amount){
        boolean isAccountFound, isBalanceSufficient;
        isBalanceSufficient = isAccountFound = false;
        try{
            File f = new File("Files/accounts.txt");
            Scanner sc= new Scanner(f);
            //verifying whether the account exists
            while(sc.hasNextLine()){
                String line = sc.nextLine();
                String[] credentials = line.split(",");
                String accNum = credentials[0];
                if(accNum.equals(senderAcc)){
                    isAccountFound = true;
                    double balance =
Double.parseDouble(credentials[3]);
                    if(credentials[1].equals("current")){ //for
current account
                        double overdraftLimit =
Double.parseDouble(credentials[2]);
                        if((overdraftLimit+balance) >=
Double.parseDouble(amount)){
                            isBalanceSufficient = true;
                        }
                    }
                    else{ // for saving account
                        if(balance >=
Double.parseDouble(amount)){
                            isBalanceSufficient = true;
                        }
                    }
                    break;
                }
            }
        }
        catch(FileNotFoundException e){
            System.out.println("An error occurred! Please try
again later");
            System.exit(0);
        }
        return new boolean[]{isAccountFound,
isBalanceSufficient};
    }
    public boolean verifyReceiverAcc(String receiverAcc){
        boolean isAccountFound = false;
        try {
            File f = new File("Files/accounts.txt");
            Scanner sc = new Scanner(f);
```



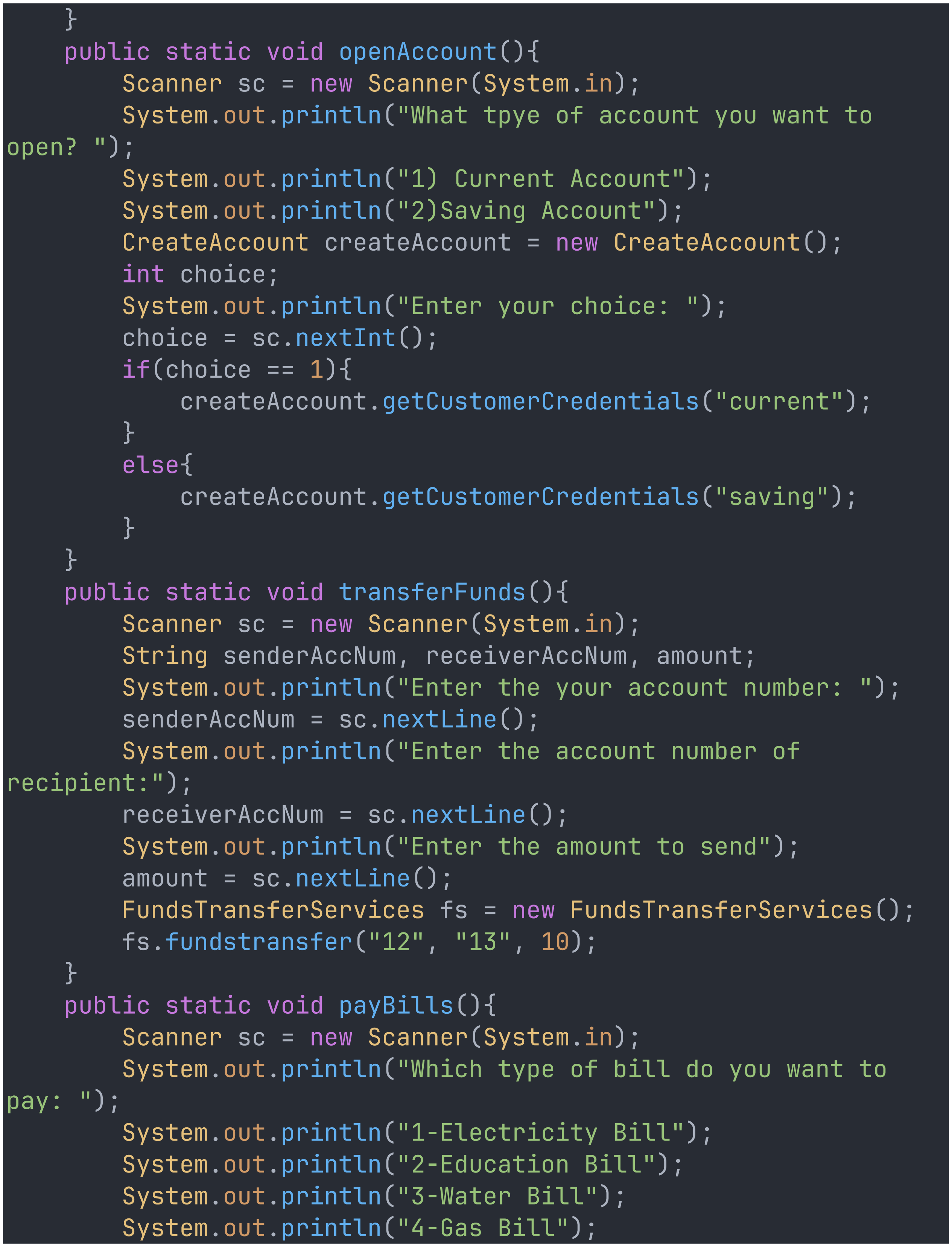
```
            //verifying whether the account exists
            while (sc.hasNextLine()) {
                String line = sc.nextLine();
                String[] credentials = line.split(",");
                String accNum = credentials[0];
                if (accNum.equals(receiverAcc)) {
                    isAccountFound = true;
                    break;
                }
            }
        }
        catch(FileNotFoundException e){
            System.out.println("An error occurred! Please try 
again later");
            System.exit(0);
        }
        return isAccountFound;
    }
    public double getBalance(String accountNumber){
        try {
            File f = new File("Files/accounts.txt");
            Scanner sc = new Scanner(f);
            //verifying whether the account exists
            while (sc.hasNextLine()) {
                String line = sc.nextLine();
                String[] credentials = line.split(",");
                String accNum = credentials[0];
                if (accNum.equals(accountNumber)) {
                    return Double.parseDouble(credentials[3]);
                }
            }
        }
        catch(FileNotFoundException e){
            System.out.println("An error occurred! Please try 
again later");
            System.exit(0);
        }
        return 0;
    }
```



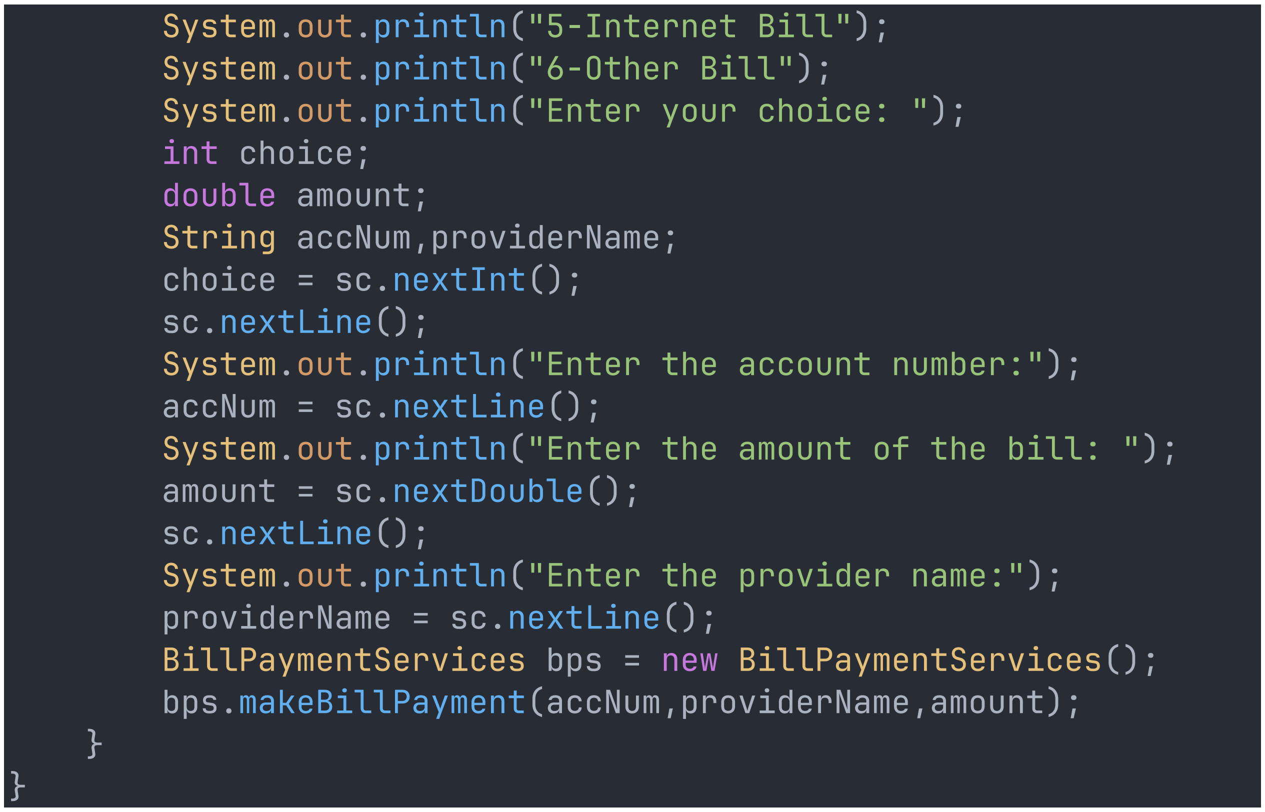
```java
    public double getOverdraftLimit(String accountNumber){
        try {
            File f = new File("Files/accounts.txt");
            Scanner sc = new Scanner(f);
            //verifying whether the account exists
            while (sc.hasNextLine()) {
                String line = sc.nextLine();
                String[] credentials = line.split(",");
                String accNum = credentials[0];
                if (accNum.equals(accountNumber)) {
                    return Double.parseDouble(credentials[2]);
                }
            }
        }
        catch(FileNotFoundException e){
            System.out.println("An error occurred! Please try
again later");
            System.exit(0);
        }
        return 0;
    }
    public String getAccType(String accNum){
        try{
            File file = new File("Files/accounts.txt");
            Scanner reader = new Scanner(file);
            while(reader.hasNextLine()){
                String line = reader.nextLine();
                String[] credentials = line.split(",");
                if(credentials[0].equals(accNum)){
                    return credentials[1];
                }
            }
            reader.close();
        }
        catch(IOException e){
            System.out.println("An error occurred! Please try
again later");
            System.exit(0);
        }
        return "";
    }
}
```

Main Class:

```
import java.util.Scanner;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("---Welcome to Online Banking System--
-");
        System.out.println("What do you want to do:");
        System.out.println("1-Open an account");
        System.out.println("2-Transfer funds");
        System.out.println("3-Pay a bill");
        System.out.println("Enter your choice: ");
        int choice;
        choice = sc.nextInt();
        switch (choice){
            case 1:
                openAccount();
                break;
            case 2:
                transferFunds();
                break;
            case 3:
                payBills();
                break;
        }
//        DateTimeFormatter dtf =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
//        LocalDateTime ldt = LocalDateTime.now();
//        System.out.println("The current locat date time is: "+
(dtf.format(ldt)).getClass().getName() );
```



```java
    }
    public static void openAccount(){
        Scanner sc = new Scanner(System.in);
        System.out.println("What tpye of account you want to
open? ");
        System.out.println("1) Current Account");
        System.out.println("2)Saving Account");
        CreateAccount createAccount = new CreateAccount();
        int choice;
        System.out.println("Enter your choice: ");
        choice = sc.nextInt();
        if(choice == 1){
            createAccount.getCustomerCredentials("current");
        }
        else{
            createAccount.getCustomerCredentials("saving");
        }
    }
    public static void transferFunds(){
        Scanner sc = new Scanner(System.in);
        String senderAccNum, receiverAccNum, amount;
        System.out.println("Enter the your account number: ");
        senderAccNum = sc.nextLine();
        System.out.println("Enter the account number of
recipient:");
        receiverAccNum = sc.nextLine();
        System.out.println("Enter the amount to send");
        amount = sc.nextLine();
        FundsTransferServices fs = new FundsTransferServices();
        fs.fundstransfer("12", "13", 10);
    }
    public static void payBills(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Which type of bill do you want to
pay: ");
        System.out.println("1-Electricity Bill");
        System.out.println("2-Education Bill");
        System.out.println("3-Water Bill");
        System.out.println("4-Gas Bill");
```



```java
        System.out.println("5-Internet Bill");
        System.out.println("6-Other Bill");
        System.out.println("Enter your choice: ");
        int choice;
        double amount;
        String accNum,providerName;
        choice = sc.nextInt();
        sc.nextLine();
        System.out.println("Enter the account number:");
        accNum = sc.nextLine();
        System.out.println("Enter the amount of the bill: ");
        amount = sc.nextDouble();
        sc.nextLine();
        System.out.println("Enter the provider name:");
        providerName = sc.nextLine();
        BillPaymentServices bps = new BillPaymentServices();
        bps.makeBillPayment(accNum,providerName,amount);
    }
}
```