CS 1083

Module 6 Assignment

By Ngoc Phuong Anh Nguyen - 3712361

July 15th 2021

Source Code:

BankAccount.java:

```
import java.io.*;
import java.text.NumberFormat;

/**
 * Basic bank account class for CS1083. Balances are not permitted to go below $0.00
 * @author Ngoc Phuong Anh Nguyen
 */
public class BankAccount implements Serializable
{
    /**
     * Account number.
     */
    private int accountNumber;
    /**
     * Account balance stored as pennies.
     */
    private int balance;

    /**
     * Constructs a BankAccount object given the account number. The initial balance is
     * set to 0 pennies, which represents $0.00
     * @param accountNumberIn The given account number.
     */
```

```
public BankAccount(int accountNumberIn)
{
    accountNumber = accountNumberIn;
    balance = 0;
}

/**
 * Constructs a BankAccount object given the account number and an initial balance.
 * @param accountNumberIn The given account number.
 * @param balanceIn The initial balance in pennies.
 */
public BankAccount(int accountNumberIn, int balanceIn)
{
    accountNumber = accountNumberIn;
    balance = balanceIn;
}

/**
 * Accessor method for accountNumber.
 * @return The account number.
 */
public int getAccountNumber()
{
    return accountNumber;
}
```

```
/**
 * Accessor method for balance.
 * @return The balance in pennies.
 */
public int getBalance()
{
    return balance;
}

/**
 * Mutator method for balance.
 * @param balanceIn The new balance in pennies.
 */
public void setBalance(int balanceIn)
{
    balance = balanceIn;
}

/**
 * Deposits money into this account.
 * @param depositAmount The amount to be deposited in pennies.
 * @throws OverpaymentException Included here only to be consistent with the
 * overridden 'deposit' method in the child LineOfCredit class.
 */
public void deposit(int depositAmount) throws OverpaymentException
{
```

```
    balance += depositAmount;
}

/**
 * Withdraws money from this account
 * @param withdrawalAmount The amount to be withdrawn in pennies. The withdrawal
 *                         happens only if the balance will remain at least $0.00.
 * @throws InsufficientFundsException When the withdrawal attempts to reduce the
 * balance below $0.00.
 */
public void withdraw(int withdrawalAmount) throws InsufficientFundsException
{
    if(balance - withdrawalAmount < 0)
    {
        throw new InsufficientFundsException();
    }
    else
    {
        balance = balance - withdrawalAmount;
    }
}

/**
 * Supplies the account number and balance as a String.
 * @return A String representation of the account number and balance for this
 * BankAccount instance. The balance is formatted in currency format
```

```
     */
    @Override
    public String toString()
    {
        NumberFormat numberFormat = NumberFormat.getCurrencyInstance();
        return "BankAccount[" +
                "accountNumber = " + accountNumber +
                ", balance = " + numberFormat.format((double)balance/100) + "]\n";
    }
}
```

UpdateAccounts.java

```
import java.io.*;
import java.text.NumberFormat;
import java.util.Scanner;
/**
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */
public class UpdateAccounts
{
    public static void main(String[] args) throws IOException
    {
        NumberFormat numberFormat = NumberFormat.getCurrencyInstance();
        final int MAX_ACCOUNT = 100;
        int bankAccountQuantity = 0;
```

```
String bankAccountRecord;
BankAccount[] bankAccounts = new BankAccount[MAX_ACCOUNT];
BankAccount bankAccount;

File inFile = new File("opening_balances.txt");
Scanner fileScanner;

fileScanner = new  Scanner(inFile);

while(fileScanner.hasNext())
{
    bankAccountRecord = fileScanner.nextLine();
    if (bankAccountQuantity < MAX_ACCOUNT)
    {
        Scanner recordScanner = new Scanner(bankAccountRecord);
        int accountNumber = recordScanner.nextInt();
        int balance = recordScanner.nextInt();

        bankAccount = new BankAccount(accountNumber,balance);
        bankAccounts[bankAccountQuantity] = bankAccount;
        bankAccountQuantity++;
    }
}
fileScanner.close();

File inFile2 = new File("transactions.txt");
```

```
fileScanner = new Scanner(inFile2);

while(fileScanner.hasNext())
{
    String transactionRecord;
    transactionRecord = fileScanner.nextLine();
    Scanner transactionScanner = new Scanner(transactionRecord);

    int position = findAccount
            (bankAccounts,bankAccountQuantity,transactionScanner.nextInt());
    String type = transactionScanner.next();
    int amount = transactionScanner.nextInt();
    try
    {
        if(type.equals("D"))
        {
            bankAccounts[position].deposit(amount);
        }
        else if(type.equals("W"))
        {
            bankAccounts[position].withdraw(amount);
        }
    }
    catch(Exception e)
    {
        System.out.println("**** " + e.getMessage() + " Exception ****\n" +
```

```
                                bankAccounts[position].toString() +
                                "Transaction type: " + type +
                                "\tTransaction Amount: " +
                                numberFormat.format((double) amount / 100) +"\n");
        }
    }
    fileScanner.close();

    PrintWriter writer;
    ObjectOutputStream ooStream = null;
    BankAccount accountFromFile;

    writer = new PrintWriter("closing_balances.txt");
    for(int i = 0; i < bankAccountQuantity; i++)
    { writer.println(bankAccounts[i].getAccountNumber()
            + "\t" + bankAccounts[i].getBalance());
    }
    writer.close();
    try
    {
        ooStream = new ObjectOutputStream(new FileOutputStream("accounts.obj"));
        fileScanner = new Scanner(new File("closing_balances.txt"));
        while (fileScanner.hasNext())
        {  accountFromFile
                = new BankAccount(fileScanner.nextInt(),//Bank Account
                fileScanner.nextInt());      // Balance
```

```
            ooStream.writeObject(accountFromFile);
        } // end while

        ooStream.flush();
    }
    catch(IOException e)
    {
        System.err.println("Error in creating object file.");
        System.exit(1);
    }
    finally
    {
        try
        {
            fileScanner.close();
            ooStream.close();

        }
        catch(IOException e)
        {
            System.err.println("Error in closing files.");
            System.exit(1);
        }
    }
    System.out.println("Files created: closing_balances.txt\n" + " " +
                       "              accounts.obj");
```

```
ObjectInputStream inStream = null;

try
{
    inStream = new ObjectInputStream(new FileInputStream("accounts.obj"));

    try
    {
        System.out.println("Accounts retrieved from accounts.obj\n" +
                           "==================================");
        while(true)
        {
            accountFromFile = (BankAccount)inStream.readObject();
            System.out.print(accountFromFile.toString());
        }
    }
    catch(EOFException e){}
}
catch(ClassNotFoundException e)
{
    System.err.println("Can't find the BankAccount class.");
    System.exit(1);
}
catch(IOException e)
{
    System.err.println("Error in reading accounts.obj");
```

```
            System.exit(1);
        }
        finally
        {
            try
            {
                inStream.close();
            }
            catch  (IOException e)
            {
                System.err.println("Error in closing accounts.obj");
            }
        }
    }

    private static int findAccount(BankAccount[] temp, int quantity, int account)
    {
        int foundPosition = -1;
        int low = 0;
        int high = quantity - 1;
        int mid;

        while(foundPosition == -1 && low <= high)
        {
            mid = (low + high) / 2;
            if(temp[mid].getAccountNumber() == account)
```

```
            {
                foundPosition = mid;
            }
            else
            {
                if(account > temp[mid].getAccountNumber())
                {
                    low = mid + 1;
                }
                else
                {
                    high = mid - 1;
                }
            }
        }
        return foundPosition;
    }
}
```

Output:

From running UpdateAccounts:

```
**** Insufficient Funds Exception ****
BankAccount[accountNumber = 3936, balance = $312.72]
Transaction type: W Transaction Amount: $395.72

**** Insufficient Funds Exception ****
BankAccount[accountNumber = 5931, balance = $279.35]
Transaction type: W Transaction Amount: $391.20

**** Insufficient Funds Exception ****
BankAccount[accountNumber = 4149, balance = $88.67]
Transaction type: W Transaction Amount: $405.25

**** Insufficient Funds Exception ****
BankAccount[accountNumber = 4149, balance = $88.67]
Transaction type: W Transaction Amount: $430.28

**** Insufficient Funds Exception ****
BankAccount[accountNumber = 4603, balance = $357.95]
Transaction type: W Transaction Amount: $441.50

**** Insufficient Funds Exception ****
BankAccount[accountNumber = 3936, balance = $33.34]
Transaction type: W Transaction Amount: $204.77
```

```
Files created: closing_balances.txt
               accounts.obj
Accounts retrieved from accounts.obj
====================================
BankAccount[accountNumber = 1324, balance = $1,323.92]
BankAccount[accountNumber = 1672, balance = $1,753.84]
BankAccount[accountNumber = 1903, balance = $561.88]
BankAccount[accountNumber = 2157, balance = $984.39]
BankAccount[accountNumber = 2519, balance = $1,415.53]
BankAccount[accountNumber = 2744, balance = $2,392.63]
BankAccount[accountNumber = 2953, balance = $545.13]
BankAccount[accountNumber = 3151, balance = $1,990.46]
BankAccount[accountNumber = 3442, balance = $739.46]
BankAccount[accountNumber = 3559, balance = $127.45]
BankAccount[accountNumber = 3936, balance = $33.34]
BankAccount[accountNumber = 4149, balance = $424.27]
BankAccount[accountNumber = 4411, balance = $718.59]
BankAccount[accountNumber = 4603, balance = $357.95]
BankAccount[accountNumber = 4848, balance = $3,016.50]
BankAccount[accountNumber = 5110, balance = $931.54]
BankAccount[accountNumber = 5236, balance = $1,235.34]
BankAccount[accountNumber = 5441, balance = $743.80]
BankAccount[accountNumber = 5581, balance = $963.66]
BankAccount[accountNumber = 5931, balance = $109.70]
```

Content of closing_balance.txt:

```
1324 132392
1672 175384
1903 56188
2157 98439
2519 141553
2744 239263
2953 54513
3151 199046
3442 73946
3559 12745
3936 3334
4149 42427
4411 71859
4603 35795
4848 301650
5110 93154
5236 123534
5441 74380
5581 96366
5931 10970
```