CS 1083

Module 5 Assignment

By Ngoc Phuong Anh Nguyen - 3712361

July 11th 2021

Source Code:

BankAccount.java:

```
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
{
    /**
     * 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 * @param withdrawalAmount The amount to be withdrawn in pennies.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(balance/100) + "]\n";
    }
}
```

LineOfCredit.java:

```
import java.text.NumberFormat;

/**
 * Line of credit class for CS1083 With a line of credit, the "balance" variable
 * represents how much money the customer has borrowed. The balance is at most $0.00,
 * meaning no money is currently owed on this account. Or the balance can be a negative
 * amount, indicating the amount currently owed.
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */
public class LineOfCredit extends BankAccount
{
    /**
     * Credit limit in pennies - a negative number, indicating the maximum amount that can
     * be borrowed on this line of credit.
     */
```

```
private int creditLimit;

/**
 * Constructs a LineOfCredit object given an account number and credit limit.
 * @param accountNumberIn The given account number.
 * @param creditLimitIn The given credit limit in pennies (a negative amount).
 */
public LineOfCredit(int accountNumberIn, int creditLimitIn)
{
    super(accountNumberIn);
    creditLimit = creditLimitIn;
}

/**
 * Constructs a LineOfCredit object given an account number, initial balance, and
 * credit limit.
 * @param accountNumberIn The given account number.
 * @param creditLimitIn The given credit limit in pennies (a negative amount).
 * @param balanceIn The given balance in pennies (0 or a negative amount)
 */
public LineOfCredit(int accountNumberIn, int creditLimitIn, int balanceIn)
{
    super(accountNumberIn,balanceIn);
    creditLimit = creditLimitIn;
}
```

```
/**
 * Accessor method for creditLimit.
 * @return The credit limit in pennies.
 */
public int getCreditLimit()
{
    return creditLimit;
}

/**
 * Mutator method for creditLimit.
 * @param creditLimitIn The given credit limit in pennies.
 */
public void setCreditLimit(int creditLimitIn)
{
    creditLimit = creditLimitIn;
}

/**
 * Deposits money into this account, which represents a payment on the amount owed.
 * The deposit is performed only if the payment does not take the balance above $0.00.
 * @param depositAmount The amount to be deposited in pennies.
 * @throws OverpaymentException When the deposit attempts to increase the balance to
 * be greater than $0.00.
 */
@Override
```

```
public void deposit(int depositAmount) throws OverpaymentException
{
    if(getBalance() + depositAmount > 0)
    {
        throw new OverpaymentException();
    }
    else
    {
        super.deposit(depositAmount);
    }
}

/**
 *
 * @param withdrawalAmount The amount to be withdrawn in pennies. The withdrawal
 *                         happens only if the balance will remain at least $0.00.
 * @throws InsufficientFundsException
 */
@Override
public void withdraw(int withdrawalAmount) throws InsufficientFundsException
{
    if(getBalance() - withdrawalAmount < creditLimit)
    {
        throw new InsufficientFundsException();
    }
    else
```

```
        {
            setBalance(getBalance() - withdrawalAmount);
        }
    }

    @Override
    public String toString()
    {
        NumberFormat numberFormat = NumberFormat.getCurrencyInstance();
        return "LineOfCredit[" +
                "accountNumber = " + getAccountNumber() +
                ", balance = " + numberFormat.format(getBalance()/100) +
                ", creditLimit = " + numberFormat.format(getCreditLimit()/100) + "]\n";
    }
}
```

OverpaymentException.java:

```
/**
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */
public class OverpaymentException extends RuntimeException
{
    public OverpaymentException()
    {
        super("Overpayment");
```

```
    }
}
```

InsufficientFundsException.java:

```
/**
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */
public class InsufficientFundsException extends RuntimeException
{
    public InsufficientFundsException()
    {
        super("Insufficient Funds");
    }
}
```

AccountTest1.java:

```
import java.text.NumberFormat;

/**
 * This is the first test driver program.
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */
public class AccountTest1
{
    public static void main(String[] args)
```

```
{
    NumberFormat numberFormat = NumberFormat.getCurrencyInstance();
    char[] transactionType = {'d', 'd', 'w', 'w'};
    int[] transactionAmount = {5000, 5000, 15000, 10000};

    System.out.println("******* Testing a bank account *******\n");

    BankAccount bankAccount = new BankAccount(1000);
    System.out.println(bankAccount);
    System.out.println(" Type      Amount   Balance\n" +
            "========  ========  ========");

    for(int i = 0; i < transactionType.length; i++)
    {
        if(transactionType[i] == 'd')
        {
            System.out.print("Deposit " +
                    Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

            bankAccount.deposit(transactionAmount[i]);
            System.out.println(Util.padLeft(numberFormat.format(
                    bankAccount.getBalance()/100),10));
        }
        else if(transactionType[i] =='w')
        {
            System.out.print("Withdraw" +
```

```
                Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

        bankAccount.withdraw(transactionAmount[i]);
        System.out.println(Util.padLeft(numberFormat.format(
                bankAccount.getBalance()/100),10));
    }
}

System.out.println("\n******* Testing a line of credit *******\n");
LineOfCredit lineOfCredit = new LineOfCredit(1001,-20000,-7500);
System.out.println(lineOfCredit);
System.out.println(" Type      Amount   Balance\n" +
        "========  ========  ========");

for(int i = 0; i < transactionType.length; i++)
{
    if(transactionType[i] == 'd')
    {
        System.out.print("Deposit " +
                Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

        lineOfCredit.deposit(transactionAmount[i]);
        System.out.println(Util.padLeft(numberFormat.format(
                lineOfCredit.getBalance()/100),10));
    }
    else if(transactionType[i] == 'w')
```

```
            {
                System.out.print("Withdraw" +
                        Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

                lineOfCredit.withdraw(transactionAmount[i]);
                System.out.println(Util.padLeft(numberFormat.format(
                        lineOfCredit.getBalance()/100),10));
            }
        }
    }
}
```

AccountTest2.java:

```
import java.text.NumberFormat;

/**
 * This is the second test driver program.
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */
public class AccountTest2
{
    public static void main(String[] args)
    {
        NumberFormat numberFormat = NumberFormat.getCurrencyInstance();
        char[] transactionType = {'d', 'd', 'w', 'w'};
```

```
int[] transactionAmount = {5000, 5000, 15000, 10000};

System.out.println("******* Testing a bank account *******\n");

BankAccount bankAccount = new BankAccount(1000);
System.out.println(bankAccount);
System.out.println(" Type      Amount   Balance\n" +
        "========  ========  ========");

for(int i = 0; i < transactionType.length; i++)
{
    if(transactionType[i] == 'd')
    {
        System.out.print("Deposit " +
                Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

        try
        {
            bankAccount.deposit(transactionAmount[i]);
            System.out.println(Util.padLeft(numberFormat.format(
                    bankAccount.getBalance()/100),10));
        }
        catch (InsufficientFundsException e)
        {
            System.out.println("("+e.getMessage()+")");
        }
```

```
    }
    else if(transactionType[i] =='w')
    {
        System.out.print("Withdraw" +
                Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

        try
        {
            bankAccount.withdraw(transactionAmount[i]);
            System.out.println(Util.padLeft(numberFormat.format(
                    bankAccount.getBalance()/100),10));
        }
        catch(InsufficientFundsException e)
        {
            System.out.println("  (" + e.getMessage() + ")");
        }
    }
}

System.out.println("\n******* Testing a line of credit *******\n");

LineOfCredit lineOfCredit = new LineOfCredit(1001,-20000,-7500);
System.out.println(lineOfCredit);
System.out.println(" Type      Amount   Balance\n" +
        "========  ========  ========");
```

```
        for(int i = 0; i < transactionType.length; i++)
        {
            if(transactionType[i] == 'd')
            {
                System.out.print("Deposit " +
                        Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

                lineOfCredit.deposit(transactionAmount[i]);
                System.out.println(Util.padLeft(numberFormat.format(
                        lineOfCredit.getBalance()/100),10));
            }
            else if(transactionType[i] == 'w')
            {
                System.out.print("Withdraw" +
                        Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

                lineOfCredit.withdraw(transactionAmount[i]);
                System.out.println(Util.padLeft(numberFormat.format(
                        lineOfCredit.getBalance()/100),10));
            }
        }
    }
}
```

AccountTest3.java:

```
import java.text.NumberFormat;

/**
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */
public class AccountTest3
{
    public static void main(String[] args)
    {
        NumberFormat numberFormat = NumberFormat.getCurrencyInstance();
        char[] transactionType = {'d', 'd', 'w', 'w'};
        int[] transactionAmount = {5000, 5000, 15000, 10000};

        System.out.println("******* Testing a bank account *******\n");

        BankAccount bankAccount = new BankAccount(1000);
        System.out.println(bankAccount);
        System.out.println(" Type      Amount   Balance\n" +
                "========  ========  ========");

        for (int i = 0; i < transactionType.length; i++)
        {
            if(transactionType[i] == 'd')
            {
                System.out.print("Deposit " +
                        Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));
```

```
    try
    {
        bankAccount.deposit(transactionAmount[i]);
        System.out.println(Util.padLeft(numberFormat.format(
                bankAccount.getBalance()/100),10));
    }
    catch (OverpaymentException e)
    {
        System.out.println("("+e.getMessage()+")");
    }
}
else if(transactionType[i] =='w')
{
    System.out.print("Withdraw" +
            Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

    try
    {
        bankAccount.withdraw(transactionAmount[i]);
        System.out.println(Util.padLeft(numberFormat.format(
                bankAccount.getBalance()/100),10));
    }
    catch(InsufficientFundsException e)
    {
        System.out.println("  (" + e.getMessage() + ")");
```

```
        }
    }
}

System.out.println("\n******* Testing a line of credit *******\n");

LineOfCredit lineOfCredit = new LineOfCredit(1001,-20000,-7500);
System.out.println(lineOfCredit);
System.out.println(" Type      Amount   Balance\n" +
        "========  ========  ========");

for (int i = 0; i < transactionType.length; i++)
{
    if(transactionType[i] == 'd')
    {
        System.out.print("Deposit " +
                Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

        try
        {
            lineOfCredit.deposit(transactionAmount[i]);
            System.out.println(Util.padLeft(numberFormat.format(
                    lineOfCredit.getBalance()/100),10));
        }
        catch (OverpaymentException e)
        {
```

```
                System.out.println("  (" + e.getMessage() + ")");
            }
        }

        else if(transactionType[i] == 'w')
        {
            System.out.print("Withdraw" +
                    Util.padLeft(numberFormat.format(transactionAmount[i]/100),10));

            try
            {
                lineOfCredit.withdraw(transactionAmount[i]);
                System.out.println(Util.padLeft(numberFormat.format(
                        lineOfCredit.getBalance()/100),10));
            }
            catch (InsufficientFundsException e)
            {
                System.out.println("  (" + e.getMessage() + ")");
            }
        }
    }
}
```

Output:

Output from AccountTest1:

```
******* Testing a bank account *******

BankAccount[accountNumber = 1000, balance = $0.00]

 Type      Amount   Balance
========  ========  ========
Deposit     $50.00    $50.00
Deposit     $50.00   $100.00
Withdraw   $150.00Exception in thread "main" InsufficientFundsException: Insufficient Funds
    at BankAccount.withdraw(BankAccount.java:89)
    at AccountTest1.main(AccountTest1.java:38)
```

Output from AccountTest2:

```
******* Testing a bank account *******

BankAccount[accountNumber = 1000, balance = $0.00]

 Type       Amount    Balance
========  ========  ========
Deposit     $50.00     $50.00
Deposit     $50.00    $100.00
Withdraw   $150.00  (Insufficient Funds)
Withdraw   $100.00      $0.00

******* Testing a line of credit *******

LineOfCredit[accountNumber = 1001, balance = -$75.00, creditLimit = -$200.00]

 Type       Amount    Balance
========  ========  ========
Deposit     $50.00    -$25.00
Deposit     $50.00Exception in thread "main" OverpaymentException: Overpayment
    at LineOfCredit.deposit(LineOfCredit.java:72)
    at AccountTest2.main(AccountTest2.java:72)
```

Output from AccountTest3:

```
******* Testing a bank account *******

BankAccount[accountNumber = 1000, balance = $0.00]

 Type       Amount    Balance
========  ========  ========
Deposit     $50.00    $50.00
Deposit     $50.00   $100.00
Withdraw   $150.00  (Insufficient Funds)
Withdraw   $100.00     $0.00

******* Testing a line of credit *******

LineOfCredit[accountNumber = 1001, balance = -$75.00, creditLimit = -$200.00]

 Type       Amount    Balance
========  ========  ========
Deposit     $50.00   -$25.00
Deposit     $50.00  (Overpayment)
Withdraw   $150.00  -$175.00
Withdraw   $100.00  (Insufficient Funds)
```