Übungen zu „Software-Entwicklung“

Ausdrücke, Anweisungen, Blöcke & Kontrollfluss

Profs. Dres. Rathke & Thies

Ihr Kunde, die SparNix Bank AG, hat festgestellt, dass sie mit der bisherigen Kontenverwaltung keine Kontoauszüge für Ihre Kunden erstellen kann. Die Kontenverwaltung, die von Ihnen implementiert wurde, kann nur den aktuellen Kontostand eines Kontos wiedergeben, verfügt aber über keinerlei Information, wie es zu diesem Kontostand gekommen ist.
Damit die SparNix Bank AG in Zukunft ihren Kunden auch Kontoauszüge zur Verfügung stellen kann, soll jede Buchung in Form einer Transaktion im jeweiligen Konto gespeichert werden und mit einem erklärenden Text und dem Datum der Buchung versehen werden können.
Der SparNix Bank AG ist auch aufgefallen, dass es unklug ist, Buchungen über zwei verschiedene Methoden zu realisieren, da die Information, ob es sich um eine Einzahlung oder eine Auszahlung handelt, schon im Vorzeichen der Summe enthalten ist. In Zukunft sollen die Methoden `deposit()` und `withdraw()` durch eine neue Methode ersetzt werden, die für jeden Buchungsvorgang eine neue Transaktion anlegt.
Das Attribut `balance` in der Klasse `Account` soll in Zukunft auch nicht mehr den aktuellen Kontostand wiedergeben, sondern den initialen (bei der Erstellung eines neuen Kontos ist dieser 0) Kontostand. Der aktuelle Kontostand lässt sich schließlich durch den initialen Kontostand und den einzelnen Transactions berechnen!
Der Wirtschaftsinformatiker Bodo Bank, ein Angestellter der SparNix Bank AG, hat für Sie schon einmal die Klasse `Transaction` erstellt, die über sämtliche Attribute und Methoden verfügt, welche die Bank für ihre zukünftige Kontenverwaltung benötigt. Ihre Aufgabe ist es nun, die Klasse `Account` soweit anzupassen, damit jede Buchung in Form einer `Transaction` darin gespeichert wird und die bisherigen Methoden mit den Transaktionen umgehen können.

In Ihrem letzten Meeting mit der SparNix Bank AG ist folgende ToDo Liste für die Weiterentwicklung des BankProjects entstanden:

1) Laden Sie sich die Datei ‘IT1_Banking_07_1.zip‘ und importieren Sie das Projekt in Eclipse.

2) Fügen Sie zur Klasse `Account` als Attribut das Array `transactions` hinzu, in dem Sie 50 Objekte vom Typ `Transaction` speichern können.

3) Überarbeiten Sie in der Klasse `Account` die Methode `getTransactions()`. Diese Methode soll als Rückgabewert das `Transaction[]` zurück geben. Überprüfen Sie Ihre Implementierung mit Hilfe der GUI. Rufen Sie hierzu im Menü ‘Übungen‘ den Menüpunkt ‘Anzahl möglicher Buchungen‘ auf.

4) Fügen Sie zur Klasse `Account` als weiteres Attribut einen `transactionPointer` vom Datentyp `int` mit dem Initialwert `0` hinzu. Dieser Pointer soll später dazu dienen, möglichst schnell auf das nächste freie Feld im Array zugreifen zu können, in dem man hier die nächste freie Position hinterlegen kann.

5) Überarbeiten Sie in der Klasse `Account` die Methode `getTransactionPointer()`. Diese Methode soll den Wert des `transactionPointer` vom Datentyp `int` zurück geben. Überprüfen Sie Ihre Implementierung mit Hilfe der GUI. Rufen Sie hierzu im Menü ‘Übungen‘ den Menüpunkt ‘Wert des Transaction-Pointers‘ auf.

6) Überarbeiten Sie in der Klasse `Account` die Methode `getCurrentBalance()`. Diese Methode soll als Rückgabewert den aktuellen Kontostand (vom Typ `double`) zurück geben. Dieser lässt sich aus dem Wert `balance` und den einzelnen Transaktionen berechnen.

7) Überarbeiten Sie in der Klasse `Account` die Methode `book(double amount, String text)`. Diese Methode hat keinen Rückgabewert und soll eine neue Instanz eines `Transaction` Objektes anlegen. Anschließend sollen dieser Instanz über die Methoden `setAmount(double amount)` und `setText(String text)` die Buchungswerte zugewiesen werden. Nach der Zuweisung der Buchungswerte soll dieses Objekt an die nächste freie Stelle im `transaction` Array gespeichert werden und der `transactionPointer` inkrementell um +1 erhöht werden. Führen Sie anschließend mit Hilfe der GUI einige Buchungen durch.

8) Überarbeiten Sie in der Klasse `Account` die Methode `updateBalance()`. Diese Methode hat keinen Rückgabewert und soll den aktuellen Kontostand (den Sie über die Methode `getCurrentBalance()` erhalten) dem Attribut `balance` zuweisen, alle Transaktionen aus dem Array entfernen und den `transactionPointer` zurück auf `0` setzen. Überprüfen Sie Ihre Implementierung mit Hilfe der GUI. Rufen Sie hierzu im Menü ‘Konto‘ den Menüpunkt ‘Kontostand aktualisieren‘ auf.

9) Korrigieren Sie in der Klasse `Account` die Methoden `getInterest()`, `isBalanceAlert()` und `isOverdrawAmount(double amount)`, so dass diese nicht mehr auf das Attribut `balance` zugreifen, sondern auf den aktuellen Kontostand! Beachten Sie die Vorzeichen bei der Methode `isOverdrawAmount(double amount)`!

10) Überarbeiten Sie in der Klasse `Account` die Methode `printAccountStatement()`. Diese Methode soll den Kontoauszug des Kontos ausgeben. Hierzu soll die `balance` zu Beginn des Auszugs ausgegeben werden, dann zu jeder Transaktion das Datum (`getDate()`), der Buchungstext (`getText()`) und der Wert der Buchung (`getAmount()`) und schließlich der aktuelle Kontostand des Kontos (`getCurrentBalance()`).Überprüfen Sie Ihre Implementierung mit Hilfe der GUI. Rufen Sie hierzu im Menü ‘Konto‘ den Menüpunkt ‘Kontostand ausgeben‘ auf.

Lösung:

```
public class Account {
    …
    /**
     * Array fuer Transactions
     */
    private Transaction[] transactions = new Transaction[50];
    /**
     * Der Transactionpointer
     */
    private int transactionPointer = 0;
    …
    /**
     * Berechnen des Zinsertrags.
     * @return Zinsertrag
     */
    public double getInterest() {
        return this.getCurrentBalance() * (interestRate / 100);
    }
    …
    /**
     * Boolsche Methode zum Abprüfen eines "kritischen"
     * Kontostandes, d.h. der Kontostand ist weniger als 1%
     * über der Kreditlinie
     */
    public boolean isBalanceAlert() {
        return (this.getCurrentBalance() <= (creditLine - (creditLine / 100)));
    }

    /**
     * Boolsche Methode zum Feststllen, ob das Abheben eins bestimmten
     * Betrags den Kontostand unterhalb der Kreditlinie führt.
```

```
 */
public boolean isOverdrawAmount(double amount) {
    return ((this.getCurrentBalance() + amount) <= creditLine);
}

/**
 * Gibt das Array mit den Transactions zurueck
 * @return transactions
 */
public Transaction[] getTransactions() {
    return this.transactions;
}

/**
 * Gibt den TransactionPointer zurueck
 * @return transactionPointer
 */
public int getTransactionPointer() {
    return this.transactionPointer;
}

/**
 * Berechnet den aktuellen Kontostand
 * @return currentBalance
 */
public double getCurrentBalance() {
    double currentBalance = this.getBalance();
    for (int i = 0; i < this.transactionPointer; i++) {
        if (this.transactions[i] != null) {
            currentBalance = currentBalance +
                    this.transactions[i].getAmount();
        }
    }
    return currentBalance;
}

/**
 * Fuegt eine Buchung hinzu
 * @param amount
 * @param text
 */
public void book(double amount, String text) {
    Transaction transaction = new Transaction();
    transaction.setAmount(amount);
    transaction.setText(text);
    this.transactions[this.transactionPointer] = transaction;
    this.transactionPointer = this.transactionPointer + 1;
}

/**
 * Aktualisiert das Attribut balance mit Hilfe der transactions
 */
public void updateBalance() {
    this.balance = this.getCurrentBalance();
    this.transactions = new Transaction[50];
    this.transactionPointer = 0;
}

/**
 * Gibt den Kontoauszug aus
 */
public void printAccountStatement() {
    Transaction tempTransaction = null;
    double tempBalance = this.getBalance();

    System.out.println("Kontoauszug fuer : " + this.getOwner().getFirstName()
            + " " + this.getOwner().getLastName());
    System.out.println("Kontonummer      : " + this.getId());
```

```
    System.out.println();
    System.out.println("Kontostand zu Beginn: " + this.getBalance());

    for (int i = 0; i < this.getTransactionPointer(); i++) {
        tempTransaction = this.getTransactions()[i];
        tempBalance = tempBalance + tempTransaction.getAmount();

        System.out.println(tempTransaction.getDate() + " " +
                tempTransaction.getText() + " " +
                tempTransaction.getAmount() + " " +
                tempBalance);
    }
}
```