# Lernziele

* Kontoverwaltung
* Es ist die Klassenstruktur sowie ein Menü zur Bedienung von zwei Konten zu implementieren.
* Zwei Konten sind für die letzte Methode überweisen notwendig
* In weiterer Folge können mehrere Konten in einer Liste angelegt werden

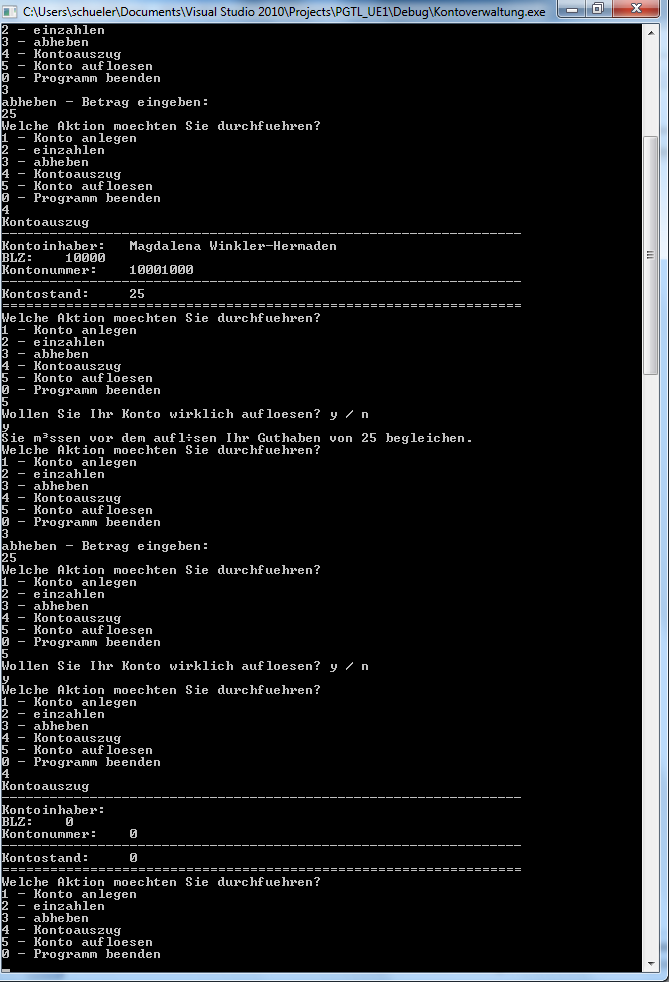
# Aufgabenstellung

Kontoklasse: (Ableitungen: Girokonto, Sparkonto, Kreditkonto)

Funktionen:

* einzahlen(), abheben(), kontoauszug()

Eigenschaften:

* Kontoinhaber
* Bankleitzahl
* Kontonummer
* Überziehungsrahmen
* Kontoführungsgebühren
* Kontostand
* Kontoart

Methoden:

* Konto anlegen
* Konto auflösen
* Einzahlen
* Abheben – kein Kreditkonto
* Kontoauszug ausgeben – Alle Daten
* (überweisen)

TODO:

* Kein Abheben bei Kreditkonto
* (überweisen)
* Kontostand ist beim Erstellen immer 0
* Kontostand aus abfrage löschen
* Abfragen ob genug geld zum überweisen auf dem konto ist
* Counter bei for schleife
* Alles kommentieren
* Dokumentation fertig schreiben
* Konto kann nur aufgeloest werden wenn der kontostand nicht im minus ist bzw kein geld darauf hat

Main Klasse (App.java)

**import** **java.util.ArrayList**;

**import** **java.util.List**;

**import** **java.util.Scanner**;

**public** **class** **App** {

**public** **static** **void** **main**(String[] args) **throws** Exception {

// fuer die endless while schleife

**boolean** isRunning = **true**;

// in dieser Liste werden alle konten die im Laufe der Programmlaufzeit erstellt werden gespeichert

List<Konto> kontos = **new** ArrayList<Konto>(**10**);

// programm laeuft so lange bis isRunning false ist

**while**(isRunning == **true**) {

// scanner definieren um Inputs lesen zu koennen

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

// menue anzeige

System.out.println(" 1 | Konto anlegen.");

System.out.println(" 2 | Konto auflösen.");

System.out.println(" 3 | Konto wählen um weitere Aktionen durchzufuehren");

System.out.println(" 4 | Programm beenden.");

// inpM stehft fuer inputMenue und speichert den Wert der beim Menue ausgewählt wurde

**int** inpM = sc.nextInt();

// switch case fuer den input vom menue

**switch**(inpM) {

// Kontotypen auswaehlen - daten eingeben - konto erstellen

**case** **1**:

System.out.println("Wählen sie den Kontotyp aus: ");

System.out.println(" 1 | Girokonto");

System.out.println(" 2 | Sparkonto ");

System.out.println(" 3 | Kreditkonto");

// kontotyp waehlen

**int** kT = sc.nextInt();

sc.nextLine();

// daten eingeben

System.out.println("Kontoinhaber: ");

String aO = sc.nextLine();

System.out.println("Bankleitzahl: ");

String blz = sc.nextLine();

System.out.println("Kontonummer: ");

String accNr = sc.nextLine();

System.out.println("Überziehlimit: ");

**float** oLimit = sc.nextFloat();

System.out.println("Kontofuehrungsgebuehren: ");

**float** fees = sc.nextFloat();

// konto erstellen

**switch**(kT) {

**case** **1**:

Girokonto gk1 = **new** Girokonto(aO, blz, accNr, oLimit, fees);

kontos.add(gk1);

**break**;

**case** **2**:

Kreditkonto kk1 = **new** Kreditkonto(aO, blz, accNr, oLimit, fees);

kontos.add(kk1);

**break**;

**case** **3**:

Sparkonto sk1 = **new** Sparkonto(aO, blz, accNr, oLimit, fees);

kontos.add(sk1);

**break**;

**default**:

**break**;

}

**break**;

// konto auswahlen - konto loeschen

**case** **2**:

**if**(kontos.size() == **0**) {

System.out.println("Es sind keine Konten vorhanden.");

} **else** {

System.out.println("Welches Konto möchten sie auflösen?");

**for**(Konto konto : kontos) {

System.out.println(kontos.indexOf(konto) + " | " + konto);

}

// input konto aufloesen

**int** inpKa = sc.nextInt();

**try** {

Konto selK = kontos.get(inpKa);

// versuchen konto zu loeschen / aufzuloesen

**if** (selK.getBalance() < **0**) {

System.out.println("Sie sind im Minus, sie können ihr Konto nicht auflösen.");

} **else** {

Konto removedK = kontos.remove(inpKa);

System.out.println("Erfolgreich aufgelöst: " + String.valueOf(removedK));

}

} **catch**(Exception e) {

System.out.println("Konto konnte nicht aufgelöst werden. Error: " + e);

}

}

**break**;

// konto waehlen - kontos anzeigen - aktion waehlen - aktion ausfuehren

**case** **3**:

**if**(kontos.size() == **0**) {

System.out.println("Es sind keine Konten vorhanden.");

} **else** {

System.out.println("Welches Konto möchten sie wählen?");

**for**(Konto konto : kontos) {

System.out.println(kontos.indexOf(konto) + " | " + konto);

}

// input konto waehlen

**int** inpKc = sc.nextInt();

**try** {

// versuchen auf das gewaehlte Konto zu wechseln - moeglicher error (index out of bound) -> deswegen try catch

Konto currK = kontos.get(inpKc);

System.out.println("Sie haben das konto: " + kontos.get(inpKc) + "ausgewählt.");

sc.nextLine();

// aktion waehlen

System.out.println(" 1 | Einzahlen.");

System.out.println(" 2 | Abheben.");

System.out.println(" 3 | Kontoauszug.");

System.out.println(" 4 | Ueberweisen.");

**int** inpC = sc.nextInt();

**switch**(inpC) {

// einzahlen

**case** **1**:

System.out.println("Wie viel möchten sie einzahlen?");

**float** payInAmount = sc.nextFloat();

currK.addBalance(payInAmount);

System.out.println("Einzahlen erfolgreich. Neuer Kontostand: " + currK.getBalance());

**break**;

// auszahlen

**case** **2**:

System.out.println("Wie viel möchten sie auszahlen?");

**float** payOutAmount = sc.nextFloat();

currK.remBalance(payOutAmount);

**break**;

// kontoauszug ausgeben

**case** **3**:

currK.bankStatement();

**break**;

// ueberweisen

**case** **4**:

// kontonummer des zu ueberweisenden kontos eingeben

System.out.println("Auf welches Konto möchten sie überweisen? (Kontonummer) ");

sc.nextLine();

String choosenKontoNr = sc.nextLine();

// boolean dient dazu um eine fehlermeldung auszugeben wenn kein Konto mit der dazugehoerigen Kontonummer gefunden wurde

**boolean** foundAccNr = **false**;

// schleife durch Kontos

**for**(Konto konto : kontos) {

// wenn die kontonummern uebereinstimmen

**if**(konto.getKontoNr().equals(choosenKontoNr)) {

foundAccNr = **true**;

System.out.println("Wie viel möchtest du ueberweisen?");

**float** betrag = sc.nextFloat();

**try** {

// versuchen dem Konto von dem ueberwiesen wird das geld abzuziehen

currK.remBalance(betrag);

// dem zu ueberweisenden Konto das geld ueberweisen

konto.addBalance(betrag);

System.out.println("Erfolgreich " + betrag + " auf das Konto von " + konto.getOwner() + " mit der Kontonummer " + konto.getKontoNr() + " ueberwiesen.");

} **catch**(Exception e){

// wenn nicht genug geld - oder ueberziehlimit zu gering usw. fehlermeldung

System.out.println(e);

}

**break**;

}

}

// wenn keine ueberinstimmende Kontonummer gefunden wurde Fehler ausgeben

**if**(!foundAccNr) {

System.out.println("Es ist ein Problem aufgetreten.");

}

foundAccNr = **false**;

**break**;

**default**:

**break**;

}

} **catch**(Exception e) {

System.out.println("Konto konnte nicht ausgewählt werden. Error: " + e);

}

}

**break**;

**case** **4**:

isRunning = **false**;

**break**;

}

}

}

}

In der Main Klasse geht es darum das ganze Programm für den User sichtbar und bedienbar zu machen.

Ich habe alles so verständlich und so kurz wie möglich versucht zu kommentieren, um hier in der Dokumentation nicht alles beschreiben zu müssen.

Ich bin mir sicher das ich noch einige Funktionalitäten vergessen habe aber alles was in der Aufgabenstellung war, funktioniert aufjedenfall.

Nun kommt die Konto Klasse:

**public** **class** **Konto** {

**private** String accOwner; // Kontoinhaber

**private** String blz; // Bankleitzahl

**private** String accNr; // Kontonummer

**private** **float** overdraftLimit; // Konto Überziehungsrahmen

**private** **float** fees; // Kontofuehrungsgebuehren

**private** **float** balance; // Kontostand

// constructor

Konto(String aO, String banklz, String aNr, **float** f) {

**this**.accOwner = aO;

**this**.blz = banklz;

**this**.accNr = aNr;

**this**.fees = f;

}

// einzahlen

**public** **void** **addBalance**(**float** amount) {

**this**.balance += amount;

}

// auszahlen - dabei ueberpruefen ob das ueberziehungslimit nicht ueberschritten wird

**public** **void** **remBalance**(**float** amount) **throws** Exception {

**if**(**this**.balance-amount > **this**.overdraftLimit) {

**this**.balance -= amount;

System.out.println("Erfolgreich abgehoben. Neuer Kontostand: " + **this**.balance);

} **else** {

**throw** **new** **Exception**("Sie können das nicht auszahlen! Überziehungslimit: " + **this**.overdraftLimit + "€. Derzeitiger Kontostand: " + **this**.balance + "€.");

}

}

// getter fuer balance

**public** **float** **getBalance**() {

**return** **this**.balance;

}

// setter fuer balance

**public** **void** **setBalance**(**float** amount) {

**this**.balance = amount;

}

//setter fuer overdraft limit

**public** **void** **setOverdraft**(**float** oLimit) {

**this**.overdraftLimit = oLimit;

}

// getter fuer Account Number

**public** String **getKontoNr**() {

**return** accNr;

}

// getter fuer account owner

**public** String **getOwner**() {

**return** accOwner;

}

// kontoauszug

**public** **void** **bankStatement**() {

System.out.println("Kontoinhaber: " + **this**.accOwner + "\nBLZ: " + **this**.blz + "\nKontonummer: " + **this**.accNr + "\nKontostand: " + **this**.balance + "€\n");

}

}

Ich muss ehrlich sagen ich habe zwischendurch in den Klassen vergessen, keine Prints zu verwenden. Sobald es aber notwendig ist werde zurückzugeben werde ich dies ändern. In der Konsole hat man es definitiv leichter, wenn man in den Klassen schon prints macht so bald bestimmte Funktionen aufgerufen werden.

Die Konto Klasse dient dazu, mal die grundsätzliche Struktur der Kontotypen (Girokonto, Sparkonto, Kreditkonto) festzulegen und dann Childs davon anzulegen.

Alles ist soweit aber mit den Kommentaren beschrieben das man weiß wofür die Funktionen sind und was sie machen.

Girokonto klasse:

**public** **class** **Girokonto** **extends** Konto{

// constructor

Girokonto(String aO, String banklz, String aNr, **float** oLimit, **float** f) {

**super**(aO, banklz, aNr, f);

**this**.setOverdraft(oLimit);

**this**.setBalance(**0**);

}

}

Im Konstruktor übergebe ich account owner, bankleitzahl, kontonummer und die fees an die Elternklasse Konto. Ich setze das Überziehlimit und die Balance auf 0. Für das überziehlimit gibt es eine eigene Funktion da ja zum Beispiel ein Sparkonto oder Kreditkonto kein überziehlimit hat und ich das mit dem einfach umgehe.

Sparkonto Klasse:

**public** **class** **Sparkonto** **extends** Konto{

// constructor

Sparkonto(String aO, String banklz, String aNr, **float** oLimit, **float** f) {

**super**(aO, banklz, aNr, f);

**this**.setBalance(**0**);

}

// da kein ueberzug moeglich ist bei einem sparkonto überlade ich die funktion set overdraft

**@Override**

**public** **void** **setOverdraft**(**float** oLimit) {

System.out.println("Not possible.");

}

**@Override**

**public** **void** **remBalance**(**float** amount) {

**if**(**this**.getBalance()-amount < **0**) {

System.out.println("Kann nicht ausgezahlt werden da sie ein Kontostand unter 0€ hätten.");

} **else** {

**this**.setBalance(**this**.getBalance()-amount);

}

}

}

Falls hier trotzdem versuch wird das überziehlimit zu setzen wird die Funktion überladen und gibt „Not possible aus“.

Hier wird auch die remBalance Funktion überladen da man nicht mehr auszahlen können sollte wenn der Kontostand dann unter 0 wäre.

Kreditkonto Klasse:

**public** **class** **Kreditkonto** **extends** Konto{

// constructor

Kreditkonto(String aO, String banklz, String aNr, **float** oLimit, **float** f) {

**super**(aO, banklz, aNr, f);

**this**.setBalance(**0**);

}

// man kann einmalig abheben aber sobald man einmal abgehoben hat und der kontostand unter 0 is kann man es nicht mehr

**@Override**

**public** **void** **remBalance**(**float** amount) {

**if**(**this**.getBalance() < **0**) {

System.out.println("no!");

} **else** {

**this**.setBalance(**this**.getBalance()-amount);

}

}

}

In der Kreditkonto Klasse überlade ich auch die Funktion remBalance, diese neue if Abfrage drinnen, stellt sicher das aus dem Kreditkonto nicht doppelt ausgezahlt wird. Sobald einmal ausgezahlt wird ist der Kontostand unter 0 und dann kann man nicht mehr auszahlen.

Und jetzt zum Output:

A computer screen with white text

Description automatically generatedProgrammstart

A screenshot of a computer

Description automatically generated 1 | Konto anlegen

A computer screen shot of white text

Description automatically generated 2 | Konto auflösen

A computer screen with white text

Description automatically generated 3 | Konto wählen für weitere Aktionen

A computer screen shot of a black screen with white text

Description automatically generated Einzahlen / Abheben auf Konto

A screenshot of a computer

Description automatically generated Kontoauszug

A screenshot of a computer program

Description automatically generated Überweisen

Es hat Spaß gemacht an diesem kleinen Projekt zu arbeiten und ich konnte viel Erfahrung sammeln, was Objektorientierung angeht. Ich habe Polymorphie gut verstanden und kann es in Zukunft anwenden. Es würde mich freuen an diesem Projekt weiterhin zu arbeiten und es zu verbessern und auszubauen. Ich werde dann die Klassen umbauen und die Konto Klasse zu einer Abstrakten Klasse machen da man ja sowieso keine Instanz von Konto erstellen kann, sondern erst von den Childs. Die Prints aus den Funktionen muss ich auch noch entfernen und dann sollte die Konsolen Applikation fertig sein.