

Lösungen Struktogramme

Hinweise:

Wir verwenden für die Aufgaben das Projekt „Girokonto“ (Abi-Projekt 2014, Aufgabe 2.3 (Girokonten, Kundenbetreuer ...)). Alle Aufgaben sind in der Klasse Kundenbetreuer durchzuführen.

+++ Aufgaben +++

Bitte **IMMER ZUERST** ein Struktogramm anfertigen, bevor Sie zu programmieren beginnen!

Alle programmierten Methoden auf korrekte Lauffähigkeit prüfen (via Startklasse).

Struktogramm und Programmcode in ein (neues, altes?) Vorlagendokument einfügen (Code als "code" formatieren), korrekte Überschriften zuweisen, zwischendurch abspeichern.

----- (A) Einsteigeraufgaben -----

A1) Programmieren Sie eine Methode `datenAusgeben(kontoinhaber:Kontoinhaber):void`, die die Daten eines Kunden ausgibt in Form

Heini Müller - Kontostand: 3000 Euro

```
+datenAusgeben(kontoinhaber:Kontoinhaber):void
    Ausgabe : kontoinhaber.getVorname() + " " + kontoinhaber.getName() + " -
    Kontostand " + kontoinhaber.getGirokonto().getKontostand() + " Euro"
```

A2) Wie A1, nur wird der String nicht **AUSGEGEBEN**, sondern **ZURÜCKGEGEBEN**. Die Methode heißt dann `datenZurueckgeben(kontoinhaber:Kontoinhaber):String`

```
+datenZurueckgeben(kontoinhaber:Kontoinhaber):String
    return kontoinhaber.getVorname() + " " + kontoinhaber.getName() + " - Kontostand: " +
    kontoinhaber.getGirokonto().getKontostand()
```

oder mit lokaler Variable, die den Rückgabewert speichert:

```
+datenZurueckgeben(kontoinhaber:Kontoinhaber):String
    rueckgabe:String = ""
    rueckgabe = rueckgabe + kontoinhaber.getVorname() + " " + kontoinhaber.getName() +
    " - Kontostand: " + kontoinhaber.getGirokonto().getKontostand()
    return rueckgabe
```

A3) Programmieren Sie eine Methode `anzahlInhaber():int`, die die Anzahl der von einem Betreuer betreuten Kontoinhaber zurückgibt.

```
+anzahlInhaber():int
return this.getKontoinhaberListe().size()
```

A4) Programmieren Sie eine Methode `listeMeinerBetreutenKunden():ArrayList<Kontoinhaber>`, die eine Liste aller betreuten Kunden ZURÜCKGIBT.

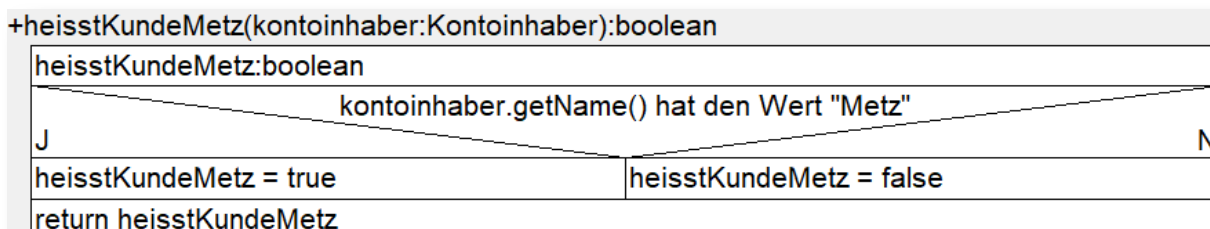
(Wenn Sie das in der Startklasse ausprobieren, benötigen Sie für die Anzeige eine `foreach`-Schleife.)

```
+listeMeinerBetreutenKunden():ArrayList<Kontoinhaber>
return this.getKontoinhaberListe()
```

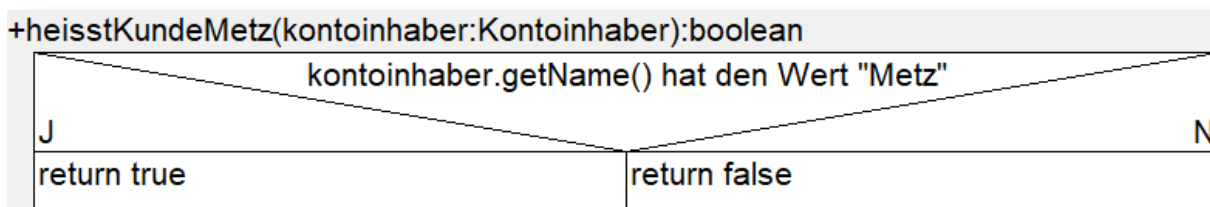
----- (B) Einfache if-Verzweigungen -----

B1) Programmieren Sie eine Methode `heisstKundeMetz(kontoinhaber:Kontoinhaber):boolean`, die prüft, ob der Nachname eines Kunden "Metz" ist. Zur Erinnerung: Verwenden Sie bei String-Vergleichen nicht `"=="`, sondern `"equals(...)"`, also bspw. `kundenname.equals("Metz")`

Möglichkeit 1: Lokale Variable verwenden, die den Rückgabewert speichert:



Möglichkeit 2: Rückgabe direkt im Bedingungsweig vornehmen.



B2) Programmieren Sie eine Methode kundePruefungAusgabe(kontoinhaber:Kontoinhaber):void, die abhängig vom Kontostand ausgibt:

Heini Müller hat mehr als 5000 Euro auf dem Konto.

oder

Heini Müller hat 5000 Euro oder weniger auf dem Konto.

+kundePruefungAusgabe (kontoinhaber:Kontoinhaber):void		
kontoinhaber.getGirokonto().getKontostand() > 5000		
J		N
Ausgabe : kontoinhaber.getVorname() + " " + kontoinhaber.getName() + " hat mehr als 5000 Euro auf dem Konto."		Ausgabe : kontoinhaber.getVorname() + " " + kontoinhaber.getName() + " hat 5000 Euro oder weniger auf dem Konto."

B3) Programmieren Sie eine Methode kundePruefung(kontoinhaber:Kontoinhaber):boolean, die abhängig vom Kontostand eines Kunden true (bei mehr als 5000 Euro) oder false (bei 5000 oder weniger) zurückgibt.

+kundePruefung(kontoinhaber:Kontoinhaber):boolean		
kontostandGroesser5000:boolean = false		
kontoinhaber.getGirokonto().getKontostand() > 5000		
J		N
kontostandGroesser5000 = true		kontostandGroesser5000 = false
return kontostandGroesser5000		

----- (C) Einfache Schleifen-----

----- (C1) For-Schleifen

C1-1) Programmieren Sie eine Methode zaehleBis(ende:int), die von 0 bis ende zählt und die Zahlen ausgibt.

+zaehleBis(ende:int):void	
zähle i:int von 0 bis ende, Schrittweite 1	
Ausgabe : i + "\n"	

C1-2) Programmieren Sie eine Methode `zaehleVonBis(start:int, ende:int)`, die von `start` bis `ende` zählt und die Zahlen ausgibt.

+zaehleVonBis(start:int, ende:int):void
zähle i:int von start bis ende, Schrittweite 1
Ausgabe : i + "\n"

C1-3) Programmieren Sie eine Methode `zaehleVonBisAbstand(start:int, ende:int, abstand:int)`, die von `start` bis `ende` zählt und die Zahlen ausgibt.

+zaehleVonBisAbstand(start:int, ende:int, abstand:int)
Anweisung
zähle i:int von start bis ende, Schrittweite abstand
Ausgabe : i + "\n"

----- (C2) While-Schleifen

C2-1) Programmieren Sie eine Methode `whileZaehlen()`, die von 0 bis 20 zählt und die Zahlen ausgibt. Verwendung einer while-Schleife.

+whileZaehlen():void
zaehler:int = 0
zaehler <= 20
Ausgabe : zaehler + "\n"
zaehler++

C2-2) Programmieren Sie eine Methode `whileZaehlenVon(start:int)`, die von `start` bis 20 zählt und die Zahlen ausgibt. Verwendung einer while-Schleife.

+whileZaehlenVon(start:int):void
start <= 20
Ausgabe : start + "\n"
start++

C2-3) Programmieren Sie eine Methode `whileZaehlenVonBis(start:int, ende:int)`, die von `start` bis `ende` zählt und die Zahlen ausgibt. Verwendung einer `while`-Schleife.

```
+whileZaehlenVonBis(start:int, ende:int):void
  start <= ende
  Ausgabe : start + "\n"
  start++
```

C2-4) Programmieren Sie eine Methode `whileZaehlenVonBisSchritt(start:int, ende:int, abstand:int)`, die von `start` bis `ende` in Schrittweite `abstand` zählt und die Zahlen ausgibt. Verwendung einer `while`-Schleife.

```
+whileZaehlenVonBisSchritt(start:int, ende:int, abstand:int)
  start <= ende
  Ausgabe : start + "\n"
  start = start + abstand
```

----- (D) Einfache foreach-Schleifen -----

D2) Kundenliste

D2-1) Programmieren Sie eine Methode `listeMeinerBetreutenKundenAlsString():void`, die eine Liste aller betreuten Kunden AUSGIBT (name, vorname, kontonummer, Kontostand). Spalten mit `\t` erzeugen.

Name	Vorname	KtoNr.	KtoStand
Schmitt	Heini	3321314	45 Euro
Müller	Bilal	587373	28348 Euro

(Es kann dabei zu Verschiebungen kommen, wenn Elemente besonders lang oder kurz sind)

```
+listeMeinerBetreutenKundenAlsString():void
  Ausgabe "Name\tVorname\tKtoNr.\tKtoStand"
  foreach(einKontoinhaber:Kontoinhaber : this.getKontoinhaberListe())
    Ausgabe : einKontoinhaber.getName() + "\t" +
    einKontoinhaber.getVorname() + "\t" +
    einKontoinhaber.getGirokonto().getKontonummer() + "\t" +
    einKontoinhaber.getGirokonto().getKontostand()
```

D2-2) Programmieren Sie eine Methode `listeMeinerBetreutenKundenAlsStringRueckgabe():String`, die eine Liste wie oben ZURÜCKGIBT. Sie müssen dazu D2-1 nur geringfügig umarbeiten.

+listeMeinerBetreutenKundenAlsStringRueckgabe():String
ausgabe:String = ""
ausgabe += "Name\tVorname\tKtoNr.\tKtoStand"
foreach(einKontoinhaber:Kontoinhaber : this.getKontoinhaberListe())
ausgabe += einKontoinhaber.getName() + "\t" + einKontoinhaber.getVorname() + "\t" + einKontoinhaber.getGirokonto().getKontonummer() + "\t" + einKontoinhaber.getGirokonto().getKontostand();
return ausgabe

D3) Wir wollen wissen, wer den höchsten Kontostand hat.

D3-1) Programmieren Sie eine Methode `reichsterMannKontostand():double`, die den Kontostand des Kontoinhabers mit dem höchsten Kontostand zurückgibt.

+reichsterMannKontostand():double
maximum:double = 0
foreach(einKontoinhaber:Kontoinhaber : this.getKontoinhaberListe())
J einKontoinhaber.getGirokonto().getKontostand() > maximum N
maximum = einKontoinhaber.getGirokonto().getKontostand()
return maximum

D3-2) Programmieren Sie eine Methode `reichsterMann():Kontoinhaber`, die den Kontoinhaber mit dem höchsten Kontostand zurückgibt.

+reichsterMannKontostand():Kontoinhaber
maximum:double = 0
k:Kontoinhaber
foreach(einKontoinhaber:Kontoinhaber : this.getKontoinhaberListe())
J einKontoinhaber.getGirokonto().getKontostand() > maximum N
k = einKontoinhaber
return k

----- (E) Kombination: Schleife/if -----

Programmieren Sie eine Methode `listeMeinerBetreutenReichenKunden():String`, die eine Liste wie in Aufgabe D2-1 ZURÜCKGIBT (als String!), allerdings sind dort nur die Kontoinhaber mit einem Kontostand von mehr als 500 Euro enthalten.

+listeMeinerBetreutenReichenKunden():String	
ausgabe:String = ""	
ausgabe += "Name\tVorname\tKtoNr.\tKtoStand"	
foreach(einKontoinhaber:Kontoinhaber : this.getKontoinhaberListe())	
<div> <div>einKontoinhaber.getGirokonto().getKontostand() > 500</div> <div> <div>J</div> <div>N</div> </div> </div>	
<div> <div>ausgabe += einKontoinhaber.getName() + "\t" + einKontoinhaber.getVorname() + "\t" + einKontoinhaber.getGirokonto().getKontonummer() + "\t" + einKontoinhaber.getGirokonto().getKontostand()</div> </div>	
return ausgabe	

----- (F) Schwierige Aufgaben -----

F1) Programmieren Sie eine Methode `reichsteKunden():ArrayList<Kontoinhaber>`, die eine Liste aller Kunden zurückgibt, deren Kontostand über dem durchschnittlichen Kontostand aller Kunden liegt.

+reichsteKunden():ArrayList<Kontoinhaber>	
// Schritt 1: durchschnittlichen Kontostand ausrechnen	
summe:double = 0	
durchschnitt:double = 0	
foreach(k:Kontoinhaber : this.getKontoinhaberListe())	
<div> <div>summe += k.getGirokonto().getKontostand()</div> </div>	
durchschnitt = summe/this.getKontoinhaberListe().size()	
// Schritt 2: reichste Kontoinhaber in neue Liste packen	
reichsteKunden:ArrayList<Kontoinhaber>	
foreach(k:Kontoinhaber : this.getKontoinhaberListe())	
<div> <div>k.getGirokonto().getKontostand() > durchschnitt</div> <div> <div>J</div> <div>N</div> </div> </div>	
<div> <div>k der Liste reichsteKunden hinzufügen</div> </div>	
return reichsteKunden	