Inhalt

(A) Einsteigeraufgaben 4
A1) datenAusgeben(kontoinhaber:Kontoinhaber):void 4
A2) datenZurueckgeben(kontoinhaber:Kontoinhaber):String 4
A3) anzahlInhaber():int 5
A4) listeMeinerBetreutenKunden():ArrayList<Kontoinhaber> 5
(B) Einfache if-Verzweigungen 6
B1) heisstKundeMetz(kontoinhaber:Kontoinhaber):boolean 6
B2) kundePruefungAusgabe(kontoinhaber:Kontoinhaber):void 7
B3) kundePruefung(kontoinhaber:Kontoinhaber):boolean 8
(C) Einfache Schleifen 9
(C1) For-Schleifen 9
C1-1) zaehleBis(ende:int) 9
C1-2) zaehleVonBis(start:int, ende:int) 9
C1-3) zaehleVonBisAbstand(start:int, ende:int, abstand:int) 10
(C2) While-Schleifen 10
C2-1) whileZaehlen() 10
C2-2) whileZaehlenVon(start:int) 11
C2-3) whileZaehlenVonBis(start:int, ende:int) 11
C2-4) whileZaehlenVonBisSchritt(start:int, ende:int, abstand:int) 12
(D) Einfache foreach-Schleifen 13
D2) Kundenliste 13
D2-1) listeMeinerBetreutenKundenAlsString():void 13
D2-2) listeMeinerBetreutenKundenAlsStringRueckgabe():String 14

D3) Reichster Mann....15
D3-1) reichsterMannKontostand():double....15
D3-2) reichsterMann():Kontoinhaber....16
(E) Kombination: Schleife/if - listeMeinerBetreutenReichenKunden():String....17
(F) Anspruchsvollere Aufgaben....18
F1) reichsteKunden():ArrayList<Kontoinhaber>....18

Aufgaben Struktogramme + Kontrollstrukturen

Durchzuführen am Beispiel der Lösung zu Abiaufgabe 2014, Aufg. 2.3 (Girokonten, Kundenbetreuer …) - s.u.

Alle Aufgaben sind in der Klasse Kundenbetreuer durchzuführen. Beachten Sie, dass im Klassendiagramm der Kundenbetreuer eine Liste "kunden" hat, die im Projekt der besseren Verständlichkeit als kontoinhaberListe benannt wurde.

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

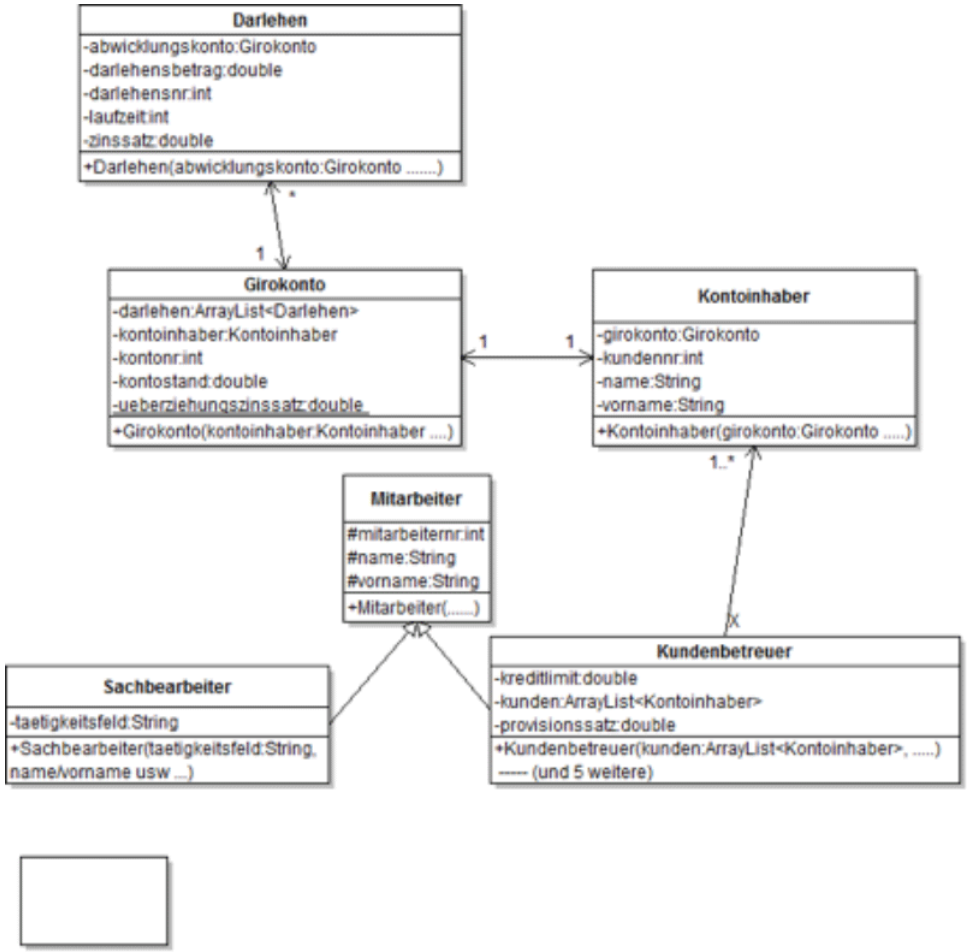
Alle programmierten Methoden auf korrekte Lauffähigkeit prüfen (via Startklasse).

Sie benötigen folgende Dateien:

zusammenfasssende_aufgaben_struktogramme_kontrollstrukturen.docx (diese Datei – hier kommen die Lösungen rein)

191126_bank – eclipse-Projekt

191126_bank/assets/KLASSENDIAGRAMM_loesungsvorschlag_abi2014_aufgabe23.png – Klassendiagramm zum Projekt (mal von einer Schülerin gemacht)



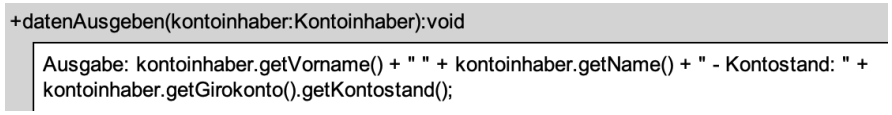
(A) Einsteigeraufgaben

A1) datenAusgeben(kontoinhaber:Kontoinhaber):void

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

Heini Müller - Kontostand: 3000 Euro

LÖSUNG: Struktogramm



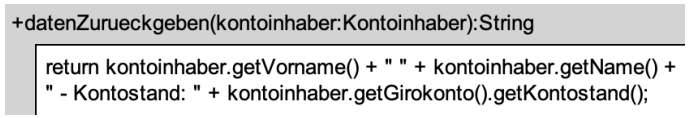
LÖSUNG: Code

```
    public void datenAusgeben(Kontoinhaber kontoinhaber) {
            System.out.println(kontoinhaber.getVorname() + " " + kontoinhaber.getName() + " - Kontostand: " +
    kontoinhaber.getGirokonto().getKontostand());
        }
```

A2) datenZurueckgeben(kontoinhaber:Kontoinhaber):String

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

LÖSUNG: Struktogramm



LÖSUNG: Code

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

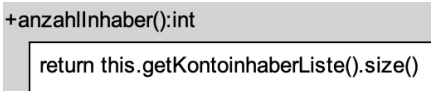
und in der Startklasse dann bspw.

```
    System.out.println(kb1.datenZurueckgeben(k3));
```

A3) anzahlInhaber():int

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

LÖSUNG: Struktogramm



LÖSUNG: Code

```
public int anzahlInhaber() {
        return this.getKontoinhaberListe().size();
    }
```

und in der Startklasse dann bspw.

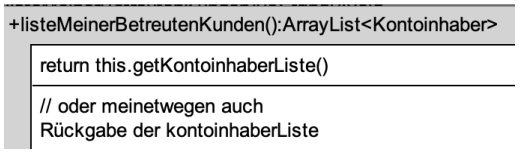
```
System.out.println("Kundenbetreuer " + kb1.getName() + " betreut " + kb1.anzahlInhaber() + " Kontoinhaber");
```

A4) listeMeinerBetreutenKunden():ArrayList<Kontoinhaber>

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.)

LÖSUNG: Struktogramm



LÖSUNG: Code

```
public ArrayList<Kontoinhaber> listeMeinerBetreutenKunden() {
        return this.getKontoinhaberListe();
    }
```

und in der Startklasse dann bspw.

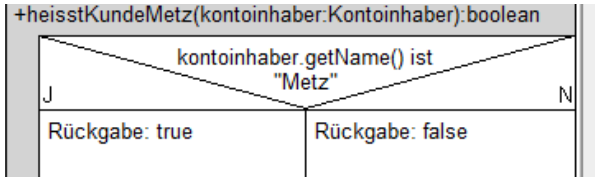
```
System.out.println("Betreute Kunden von " + kb1.getName() + ":");
        for(Kontoinhaber einKontoinhaber : kb1.listeMeinerBetreutenKunden()) {
        System.out.println(einKontoinhaber.getName());
        }
```

(B) Einfache if-Verzweigungen

B1) heisstKundeMetz(kontoinhaber:Kontoinhaber):boolean

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. kundename.equals("Metz")

LÖSUNG: Struktogramm



LÖSUNG: Code

```
public boolean heisstKundeMetz(Kontoinhaber kontoinhaber) {
    if(kontoinhaber.getName().equals("Metz")) {
        return true;
    }
    else {
        return false;
    }
}
```

B2) kundePruefungAusgabe(kontoinhaber:Kontoinhaber):void

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.

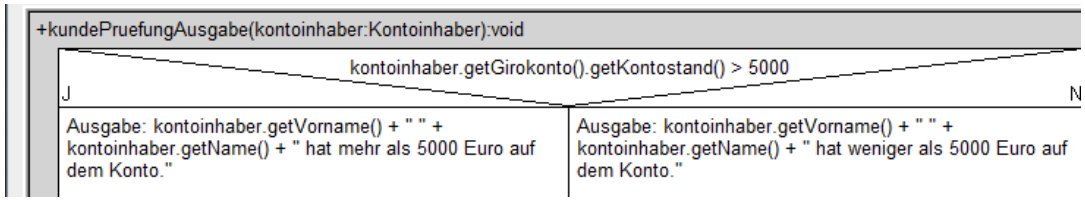


Bild hier einfügen

LÖSUNG: Code

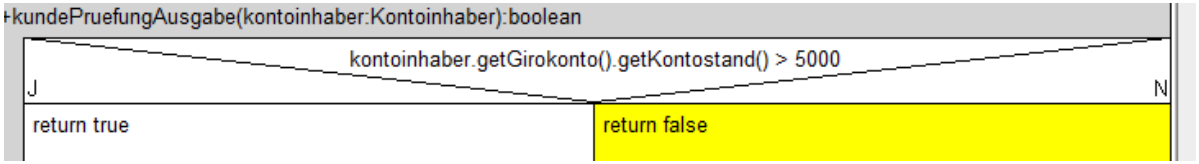
```
    public void kundePruefungAusgabe(Kontoinhaber kontoinhaber) {
            if(kontoinhaber.getGirokonto().getKontostand() > 5000) {
                  System.out.println(kontoinhaber.getVorname() + " " + kontoinhaber.getName() + " hat mehr als 5000
    Euro auf dem Konto.");
            }
            else {
                  System.out.println(kontoinhaber.getVorname() + " " + kontoinhaber.getName() + " hat weniger als 5000
    Euro auf dem Konto.");

            }
        }
```

B3) kundePruefung(kontoinhaber:Kontoinhaber):boolean

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.

LÖSUNG: Struktogramm



LÖSUNG: Code

```
public boolean kundePruefung(Kontoinhaber kontoinhaber) {
        if(kontoinhaber.getGirokonto().getKontostand() > 5000) {
              return true;
        }
        else {
              return false;
        }
    }
```

(C) Einfache Schleifen

(C1) For-Schleifen

C1-1) zaehleBis(ende:int)

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

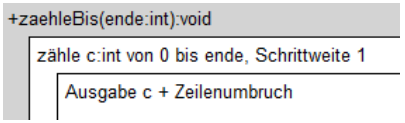


Bild hier einfügen

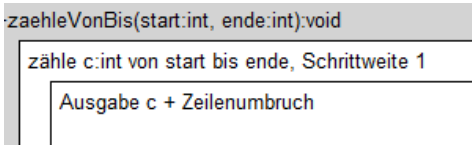
LÖSUNG: Code

```
public void zaehleBis(int ende) {
  for(int c = 0;c <= ende;c++) {
        System.out.println(c);
  }
}
```

C1-2) zaehleVonBis(start:int, ende:int)

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

LÖSUNG: Struktogramm



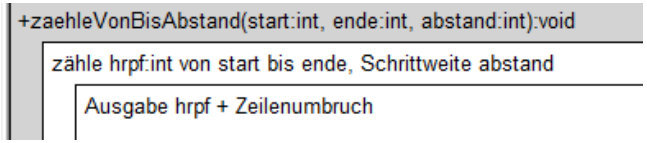
LÖSUNG: Code

```
public void zaehleVonBis(int start, int ende) {
  for(int c = start;c <= ende;c++) {
        System.out.println(c);
  }
}
```

C1-3) zaehleVonBisAbstand(start:int, ende:int, abstand:int)

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

LÖSUNG: Struktogramm



LÖSUNG: Code

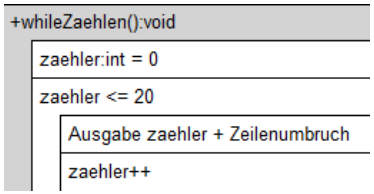
```
public void zaehleVonBisAbstand(int start, int ende, int abstand) {
  for(int hrpf = start;hrpf <= ende;hrpf = hrpf + abstand) {
        System.out.println(hrpf);
  }
}
```

(C2) While-Schleifen

C2-1) whileZaehlen()

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

LÖSUNG: Struktogramm

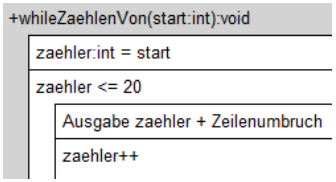


LÖSUNG: Code

```
public void whileZaehlen() {
  int zaehler = 0;
  while(zaehler <= 20) {
        System.out.println(zaehler);
        zaehler++;
  }
}
```

C2-2) whileZaehlenVon(start:int)

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

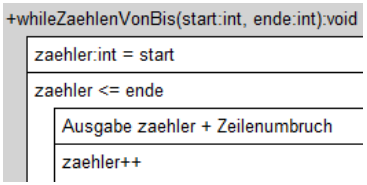


LÖSUNG: Code

```
public void whileZaehlenVon(int start) {
  int zaehler = start;
  while(zaehler <= 20) {
        System.out.println(zaehler);
        zaehler++;
  }
}
```

C2-3) whileZaehlenVonBis(start:int, ende:int)

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



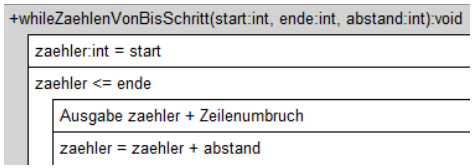
LÖSUNG: Code

```
public void whileZaehlenVonBis(int start, int ende) {
  int zaehler = start;
  while(zaehler <= ende) {
        System.out.println(zaehler);
        zaehler++;
  }
}
```

C2-4) whileZaehlenVonBisSchritt(start:int, ende:int, abstand:int)

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.

LÖSUNG: Struktogramm



LÖSUNG: Code

```
public void whileZaehlenVonBisSchritt(int start, int ende, int abstand) {
  int zaehler = start;
  while(zaehler <= ende) {
        System.out.println(zaehler);
        zaehler = zaehler + abstand;
  }
}
```

(D) Einfache foreach-Schleifen

D2) Kundenliste

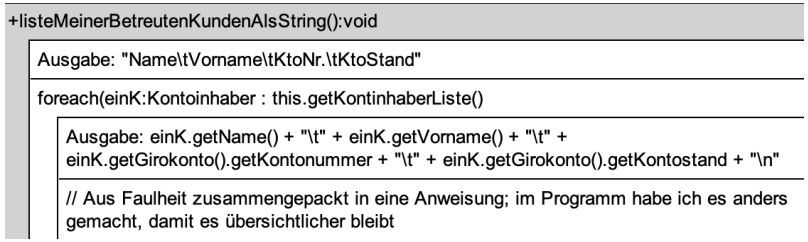
D2-1) listeMeinerBetreutenKundenAlsString():void

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)

LÖSUNG: Struktogramm



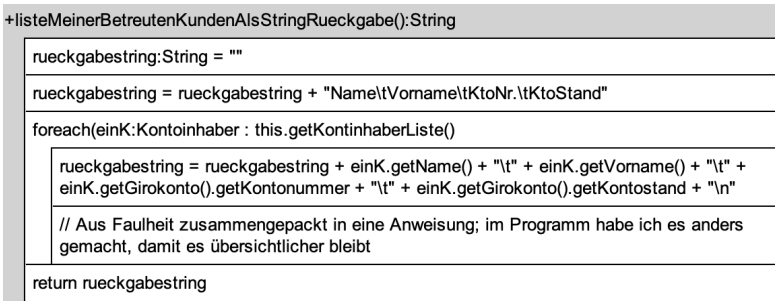
LÖSUNG: Code

```
    public void listeMeinerBetreutenKundenAlsString() {
            System.out.println("Name\tVorname\tKtoNr.\tKtoStand");
            for(Kontoinhaber einK : this.getKontoinhaberListe()) {
                  System.out.print(einK.getName() + "\t");
                  System.out.print(einK.getVorname() + "\t");
                  System.out.print(einK.getGirokonto().getKontonummer() + "\t");
                  System.out.print(einK.getGirokonto().getKontostand() + "\t");
                  System.out.print("\n");
            }
      }
```

D2-2) listeMeinerBetreutenKundenAlsStringRueckgabe():String

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

LÖSUNG: Struktogramm



LÖSUNG: Code

```
    public String listeMeinerBetreutenKundenAlsStringRueckgabe() {
            String rueckgabestring = "";
            rueckgabestring = rueckgabestring + "Name\tVorname\tKtoNr.\tKtoStand";
            for(Kontoinhaber einK : this.getKontoinhaberListe()) {
                  rueckgabestring = rueckgabestring + einK.getName() + "\t";
                  rueckgabestring = rueckgabestring + einK.getVorname() + "\t";
                  rueckgabestring = rueckgabestring + einK.getGirokonto().getKontonummer() + "\t";
                  rueckgabestring = rueckgabestring + einK.getGirokonto().getKontostand() + "\t";
                  rueckgabestring = rueckgabestring + "\n";
            }
            return rueckgabestring;
        }
```

und in der Startklasse dann bspw.

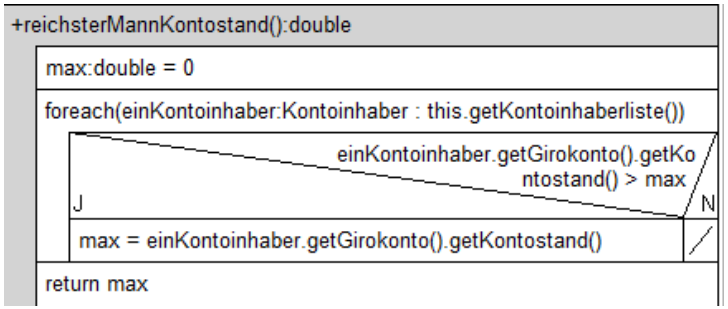
```
            System.out.println(kb1.listeMeinerBetreutenKundenAlsStringRueckgabe());
```

D3) Reichster Mann

D3-1) reichsterMannKontostand():double

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

LÖSUNG: Struktogramm



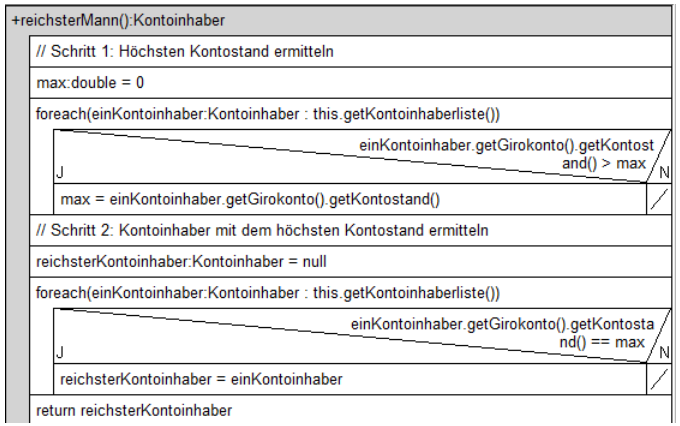
LÖSUNG: Code

```
public double reichsterMannKontostand() {
  double max = 0;
  for (Kontoinhaber einKontoinhaber : this.getKontoinhaberListe()) {
        if(einKontoinhaber.getGirokonto().getKontostand() > max) {
              max = einKontoinhaber.getGirokonto().getKontostand();
        }
  }
  return max;
}
```

D3-2) reichsterMann():Kontoinhaber

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

LÖSUNG: Struktogramm



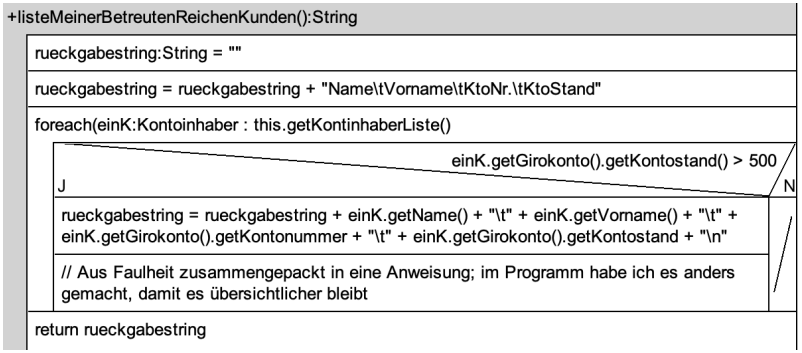
LÖSUNG: Code

```
public Kontoinhaber reichsterMann() {
  double max = 0;
  for (Kontoinhaber einKontoinhaber : this.getKontoinhaberListe()) {
        if(einKontoinhaber.getGirokonto().getKontostand() > max) {
              max = einKontoinhaber.getGirokonto().getKontostand();
        }
  }
  Kontoinhaber reichsterKontoinhaber = null;
  for (Kontoinhaber einKontoinhaber : this.getKontoinhaberListe()) {
        if(einKontoinhaber.getGirokonto().getKontostand() == max) {
              reichsterKontoinhaber = einKontoinhaber;
        }
  }
  return reichsterKontoinhaber;
}
```

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

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.

LÖSUNG: Struktogramm



LÖSUNG: Code

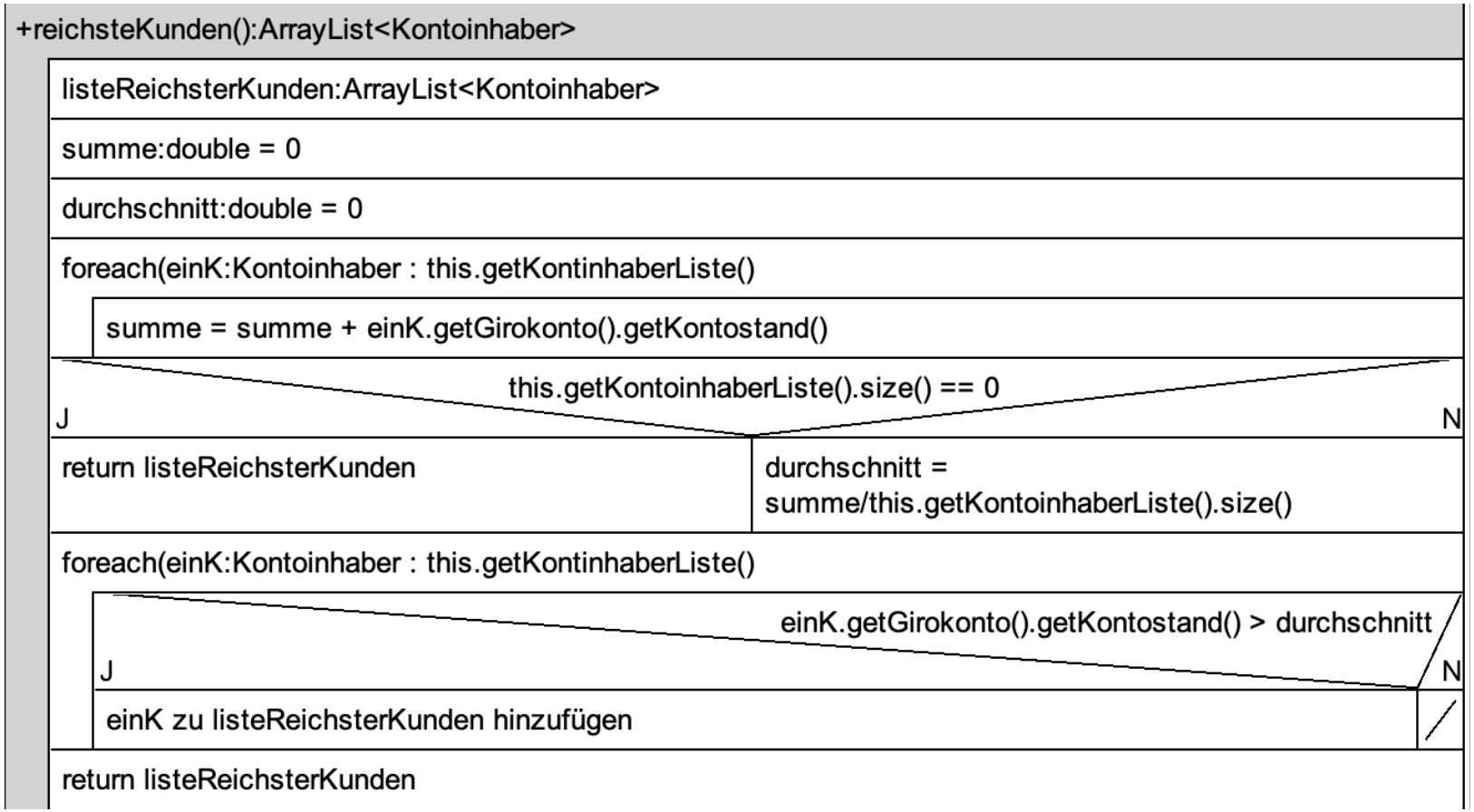
```
    public String listeMeinerBetreutenReichenKunden() {
            String rueckgabestring = "";
            rueckgabestring = rueckgabestring + "Name\tVorname\tKtoNr.\tKtoStand";
            for(Kontoinhaber einK : this.getKontoinhaberListe()) {
                  if(einK.getGirokonto().getKontostand() > 500) {
                  rueckgabestring = rueckgabestring + einK.getName() + "\t";
                  rueckgabestring = rueckgabestring + einK.getVorname() + "\t";
                  rueckgabestring = rueckgabestring + einK.getGirokonto().getKontonummer() + "\t";
                  rueckgabestring = rueckgabestring + einK.getGirokonto().getKontostand() + "\t";
                  rueckgabestring = rueckgabestring + "\n";
                  }
            }
            return rueckgabestring;
        }
```

(F) Anspruchsvollere Aufgaben

F1) reichsteKunden():ArrayList<Kontoinhaber>

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

LÖSUNG: Struktogramm



LÖSUNG: Code

```
        public ArrayList<Kontoinhaber> reichsteKunden() {
            // Liste erstellen, in die wir die reichsten Kunden packen und die wir dann zurückgeben
            ArrayList<Kontoinhaber> listeReichsterKunden = new ArrayList<Kontoinhaber>();
            // erst mal durchschnittlichen Kontostand ermitteln
            double summe = 0;
            double durchschnitt = 0;
            for(Kontoinhaber einK : this.getKontoinhaberListe()) {
                  summe += einK.getGirokonto().getKontostand();
            }
            if(this.getKontoinhaberListe().size() == 0) {
                  return listeReichsterKunden; // leere liste, wenn einer keine kunden hat
            }
            else {
            durchschnitt = summe/this.getKontoinhaberListe().size();
            // waere die liste des Kundenbetreuers leer, würden wir jetzt eine Fehlermeldung erhalten (division by
    zero)
            // deshalb packen wir das in eine if-abfrage.
            // wenn Sie motiviert sind, können sie schon mal surfen, wie man das mit try-catch lösen
            // würde - machen wir nämlich noch.
            }

            // jetzt alle, die über dem durchschnitt liegen, in die liste packen
            for(Kontoinhaber einK : this.getKontoinhaberListe()) {
                  if(einK.getGirokonto().getKontostand() > durchschnitt) {
                        listeReichsterKunden.add(einK);
                  }
            }
            return listeReichsterKunden;
        }
```

und in der Startklasse dann bspw.

```
System.out.println("Reichste Kunden von Betreuer " + kb1.getName() + ":");
        for(Kontoinhaber einReicherKunde : kb2.reichsteKunden()) {
        System.out.println(einReicherKunde.getName());
        }
```