Praktikum 8

Realisieren Sie eine Anwendung mit Bankkonten! Schreiben Sie in einer separaten Textdatei mit, welche Teilaufgaben Sie bereits bearbeitet haben. Somit wissen Sie in der nächsten Stunde an welcher Stelle Sie weiterarbeiten müssen.

1. Legen Sie eine globale abstrakte Klasse Bankkonto an (ZXXX_XX_CL_BANKKONTO):
   a) Jedes Bankkonto hat eine Kontonummer, einen Kontostand und einen Kontoinhaber. Diese Komponenten sind alle in einer Struktur zusammengefasst. Definieren Sie dafür innerhalb der Klasse einen Strukturtyp TS_BANKKONTO:
      - `kontonummer(10) TYPE n`
      - `kontostand(8)   TYPE p DECIMALS 2`
      - `kontoinhaber    TYPE string`

      Hinweis: Im OO-Kontext sind bei den Typen c, p und n explizite Längenangaben erforderlich, sonst bekommt man eine Fehlermeldung. Die gültige Länge für Typ p liegt zwischen 1 und 16 Byte. In Klammern muss diese Länge hier angegeben werden (wie auch bei Typ n). Die Standardlänge von Typ ist 8 Byte und es können dann Zahlen von -10^{13} bis $+10^{13}$ abgespeichert werden.
      (Noch genauer kann man das in der ABAP-Schlüsselwortdokumentation nachlesen.)

   b) Legen Sie anschließend das private Attribut MS_BANKKONTO an.
   c) Legen Sie außerdem den Typ TN_GELDBETRAG in der Klasse an (2 Nachkommastellen).
   d) Bei der Erzeugung eines Kontos soll die Kontonummer im Hintergrund automatisch fortlaufend vergeben werden, beginnend bei 0100000001. Der Kontostand wird initial immer auf 5 € gesetzt. Es muss beim Konstruktor also nur der Name des Kontoinhabers übergeben werden.
      Hinweis: Für die fortlaufende Nummer braucht die Klasse ein weiteres geeignetes Attribut, dass für jedes Konto erhöht werden muss. Für dieses Attribut verwenden Sie den Datentyp TS_BANKKONTO-KONTONUMMER.
   e) Jedes Konto kann Auskunft über seine Kontonummer (GET_KONTONR) und seinen Kontostand (GET_KONTOSTAND) geben. Legen Sie dafür funktionale Methoden an, die die entsprechenden Werte zurückgeben. Für die Parameter verwenden Sie TS_BANKKONTO-KONTONUMMER bzw. TS_BANKKONTO-KONTOSTAND als Datentyp.
   f) Zum Ändern des Kontostandes existiert die Methode SET_KONTOSTAND. Die Methode kann nur von Unterklassen aufgerufen werden und der aktuelle Kontostand wird durch den übergebenen Kontostand ersetzt.
   g) Jedes Konto kann seine Details ausgeben (GET_DATA). Diese Daten werden als string zurückgegeben. Vorerst wird dabei nur der Name des Kontoinhabers zurückgegeben.
   h) Eine abstrakte Methode ABHEBEN ermöglicht das Abheben eines Geldbetrags vom Bankkonto. Der Methode muss der Betrag übergeben werden, der abgehoben werden soll (Datentyp TN_GELDBETRAG verwenden).
   i) Eine Methode EINZAHLEN ermöglicht das Einzahlen eines Geldbetrags auf das Bankkonto. Der eingezahlte Betrag muss dafür übergeben werden und wird zum aktuellen Kontostand

addiert. Achtung: die Methode ist nicht abstrakt. Nach dem Einzahlen wird mit WIRTE folgender Text ausgegeben: „Es wurden <betrag> € erfolgreich in das Konto <kontonummer> eingezahlt.“.

j) Man soll eine Überweisung an ein anderes Konto tätigen können. Dafür steht die Methode TRANSFER zur Verfügung. Als Information muss man dieser Methode also den Betrag übergeben sowie das Bankkonto, auf das der Betrag überwiesen werden soll. Ein der Import-Parameter muss also das Objekt eines anderen Bankkontos sein (siehe Screenshoot unten). Heben Sie den Betrag vom aktuellen Konto ab und zahlen den Betrag in das andere Konto ein. Mit Hilfe von WRITE wird folgender Text ausgegeben: „Transfer von <betrag> € von Kontonr <kontonr> nach Kontonr <andere_kontonr>“. Vor dem Abheben bzw. nach dem Einzahlen wird der Befehl ULINE ausgeführt.

IN_BETRAG	Importing	☐	☐	Type	TS_BANKKONTO-KONTOSTAND
IO_BANKKONTO	Importing	☐	☐	Type Ref To	ZXXX_TP_CL_BANKKONTO

2. Legen Sie eine globale Klasse Girokonto an (ZXXX_XX_CL_GIROKONTO):
   a) Für ein Girokonto wird bei der Eröffnung ein Dispokredit festgelegt. Dieses zusätzliche Attribut hat den Datentyp TN_GELDBETRAG und muss beim Konstruktor ebenfalls übergeben werden.
   b) Eine Abhebung ist nur möglich, wenn dadurch der Dispokredit nicht überschritten wird. Ist ein Abheben möglich, dann wird mit WRITE die Meldung „Es wurden <betrag> € erfolgreich vom Konto <kontonummer> abgehoben.“. Ansonsten wird die Meldung „Dieser Betrag kann vom Konto <kontonr> nicht abgehoben werden. Der Dispokredit von <dispokredit> € kann nicht überschritten werden.“ ausgegeben.
   Beispiel: Der Dispokredit ist 1000 € und man hat noch 100 € am Konto. Möchte man dann 1200 € abheben, dann wäre man bereits 1100 € im Minus und der Dispokredit ist überschritten.
   c) Ein Girokonto gibt die Details wie folgt zurück (Methode GET_DATA): „Girokonto von <kontoinhaber>, Kontonummer <kontonummer>, aktueller Kontostand <kontostand> €, Disporahmen: <dispokredit> €“

3. Legen Sie eine globale Klasse Sparkonto an (ZXXX_XX_CL_SPARKONTO).
   a) Die Erstellung eines Sparkontos erzwingt die initiale Einzahlung eines Betrags. D.h. beim Konstruktor muss zusätzlich ein Geldbetrag übergeben werden, der beim Anlegen eines Sparkontos eingezahlt wird. Verwenden Sie dafür die Methode EINZAHLEN.
   b) Ein Sparkonto erlaubt keine Überziehung. D.h. beim Abheben darf der Kontostand nie negativ werden. Im Erfolgsfall wird die gleiche Meldung wie beim Girokonto ausgegeben. Ansonsten erscheint die Meldung „Eine Überziehung des Kontos <kontonr> ist nicht erlaubt. Abheben von <betrag> € daher nicht möglich.“
   c) Für die Redefinition von GET_DATA orientieren Sie sich am Screenshoot auf der nächsten Seite (ganz unten die letzten Zeilen).

4. Legen Sie ein Programm ZXXX_XX_PROG_BANKKONTO an, um Ihre Klassen zu testen:
   a) Legen Sie zwei Girokonten an:
   Florian Maier mit Dispokredit 1000 € und Sabine Huber mit Dispokredit 2000 €
   b) Legen Sie zwei Sparkonten an:
   Michael Oswald (Einzahlung 3150,49 €) und Franziska Loibl (Einzahlung 5005,90 €)
   c) Fügen Sie alle Bankkonten zu einer internen Tabelle hinzu. Zahlen Sie in jedes Konto einen Betrag von 200 € ein und geben anschließend nur die Kontonummer und den Kontostand für jedes Bankkonto aus.
   d) Führen sie für das Sparkonto von Franziska Loibl zwei Abhebungen von zuerst 4000 € und anschließend von 2000 € durch. Hinweis: mit gt_konten[4] können Sie auf das Objekt mit Index 4 zugreifen.
   e) Führen Sie eine Überweisung über 350 € vom Girokonto von Florian Maier zum Sparkonto von Michael Oswald durch.
   f) Geben Sie mit Hilfe von GET_DATA die Details von allen Bankkonten aus.

Bei der Ausgabe am Ende sollte sich folgendes Bild ergeben:

```
Es wurden 3150.49 € erfolgreich in das Konto 0100000003 eingezahlt.
Es wurden 5005.90 € erfolgreich in das Konto 0100000004 eingezahlt.

In jeds Konto werden 200 € eingezahlt:
Es wurden 200.00 € erfolgreich in das Konto 0100000001 eingezahlt.
0100000001 : 205,00 €
Es wurden 200.00 € erfolgreich in das Konto 0100000002 eingezahlt.
0100000002 : 205,00 €
Es wurden 200.00 € erfolgreich in das Konto 0100000003 eingezahlt.
0100000003 : 3.355,49 €
Es wurden 200.00 € erfolgreich in das Konto 0100000004 eingezahlt.
0100000004 : 5.210,90 €

Es wurden 4000.00 € erfolgreich vom Konto 0100000004 abgehoben.
Eine Überziehung des Kontos 0100000004 ist nicht erlaubt. Abheben von 2000.00 € daher nicht möglich.

Transfer von 350.00 € von Kontonr 0100000001 nach Kontonr 0100000003
```

```
Es wurden 350.00 € erfolgreich vom Konto 0100000001 abgehoben.
Es wurden 350.00 € erfolgreich in das Konto 0100000003 eingezahlt.
```

```
Girokonto von Florian Maier, Kontonummer 0100000001, aktueller Kontostand -145.00 €, Disporahmen: 1000.00 €
Girokonto von Sabine Huber, Kontonummer 0100000002, aktueller Kontostand 205.00 €, Disporahmen: 2000.00 €
Sparkonto Nr. 0100000003 von Michael Oswald, aktueller Kontostand 3705.49 €.
Sparkonto Nr. 0100000004 von Franziska Loibl, aktueller Kontostand 1210.90 €.
```