Prozeduren, Algorithmen, Pseudocode - Aufgaben

Aufgabe 1

Die Systemsoft GmbH soll für die Media-HO GmbH eine Prozedur entwickeln, die in Sekunden erfasste Zeiten im Format Wochen : Tage : Stunden : Minuten : Sekunden darstellt.

Beispiel: 788.645 Sekunden = 1 Woche, 2 Tage, 3 Stunden, 4 Minuten, 5 Sekunden

Die Zeitwerte des Formats (Wochen, Tage, Stunden, Minuten, Sekunden) sollen in einem statischen, eindimensionalen Datenfeld an die aufrufende Prozedur zurückgegeben werden.

Schreiben Sie diese Prozedur in Pseudocode oder in einer gebräuchlichen Programmiersprache.

Lösung

```
Option Explicit
Option Base 0

Public Sub zTeiler(datenfeld() As Double, zeit As Double)

Dim temp As long, rest As long, woche As Integer, tag As Integer, stunde As Integer, minute As Integer, sekunde As Integer, i As Integer

'Woche ermitteln
temp = zeit \ 604800
woche = Fix(temp)
rest = zeit Mod 604800

'Tage ermitteln
temp = rest \ 86400
tag = Fix(temp)
rest = zeit Mod 86400

'Stunden ermitteln
temp = rest \ 3600
stunde = Fix(temp)
rest = zeit Mod 3600

'Minuten und (Rest-)Sekunden ermitteln
temp = rest \ 60
minute = Fix(temp)
sekunde = zeit Mod 60

'Datenfeld erstellen und füllen
ReDim datenfeld(0)
ReDim Preserve datenfeld(5)
datenfeld(1) = woche
datenfeld(2) = tag
datenfeld(3) = stunde
datenfeld(4) = minute
datenfeld(5) = sekunde

End Sub

Sub zHaupt()
Dim meinDatenfeld() As Double
Dim i As Integer
Dim sek_zeit As Double
sek_zeit = 788645

Call zTeiler(meinDatenfeld(), sek_zeit)

For i = 1 To 5
Debug.Print meinDatenfeld(i)
Next i

End Sub
```

Aufgabe 2

Die Brück & Saar GmbH soll für die AllSolar GmbH ein Programm erstellen, mit dem der wirtschaftliche Erfolg einer Solaranlage berechnet werden kann.

a)

PLZ	Sonnenstunden
...	...
50606	1.200
51491	1 '100
...	...

Erstellen Sie einen Algorithmus für die Funktion holeSonnenstunden(), welche die Sonnenstunden eines Postleitzahlbereichs aus dem zweidimensionalen Array Sonnenstunden liefert. Das Übergabeparameter ist eine PLZ.
(Darstellung in Pseudocode, PAP oder Struktogramm) (10 Punkte)

Hinweis:

- Die Zeilen des Arrays sind nach Postleitzahlen aufsteigend sortiert.
- Wird die übergebene PLZ im Array nicht gefunden, so soll die Sonnenstundenzahl der nächstkleineren PLZ verwendet werden.
- Ist die eingegebene PLZ kleiner die kleinste im Array vorhandene PLZ, wird -1 zurückgegeben.

Lösung

```
Funktion holeSonnenstunden(plz: Integer): Integer

Lokale Variable: erg: Integer

erg = -1
für i = 0, 1, Anzahl Zeilen Sonnenstunden -1
    wenn plz = Sonnenstunden[i, 0]
        erg = Sonnenstunden[i, 1]
        i = Anzahl Zeilen Sonnenstunden
    sonst
        wenn plz < Sonnenstunden[i, 0]
            i = Anzahl Zeilen Sonnenstunden
        sonst
            erg = Sonnenstunden[i, 1]
        ende wenn
    ende wenn
ende für

Rückgabe erg

Ende Funktion
```

Aufgabe 3

b) Erstellen Sie einen Algorithmus zur Ermittlung der Gesamtvergütung in Euro.
(Darstellung in Pseudocode, PAP oder Struktogramm) (15 Punkte)

Übergabeparameter:	Beschreibung einer Solar Anlage (Beispiel)
Nennleistung	Standort: Köln (PLZ 50606)
PLZ	Jährliche Sonnenstunden: 1.200
Abweichung von Süden/Grad	Nennleistung bei direkter Südausrichtung: 50.000 kWh/kWp
Laufzeit	Ausrichtung: 31 Grad Abweichung von Süden
Vergütung je kWh	Anschaffungskosten: 20.000,00 €
	Betriebsdauer: 20
	Vergütung je kWh: 0,45 €

Hinweis:

- Die Nennleistung wird nur bei einer Ausrichtung nach Süden und bei 1.800 Std. Sonneneinstrahlung erreicht.
- Je Grad Abweichung von der Südausrichtung verändert sich die Nennleistung um 0,5 %.
- Die Nennleistung einer Solaranlage nimmt je Betriebsjahr linear um 2 % ab.

Lösung

```
Funktion ermittleGesamtverguetung(
    nennleistung: Integer
    plz: Integer
    grad: Double
    laufzeit: Integer
    vergütung: Double)

Lokale Variablen:   summe: Double
                    faktor: Double
                    ertrag: Double
                    sonnenstd: Integer

summe = 0
faktor = 1.0
sonnenstd = holeSonnenstunden(plz)

für i = 1, 1, laufzeit
    ertrag = nennleistung * sonnenstd/1800 * (1-grad * 0,005) * faktor * vergütung * sonnenstd
    summe = summe + ertrag
    faktor = faktor – 0,02
ende für

Rückgabe summe

Ende Funktion
```

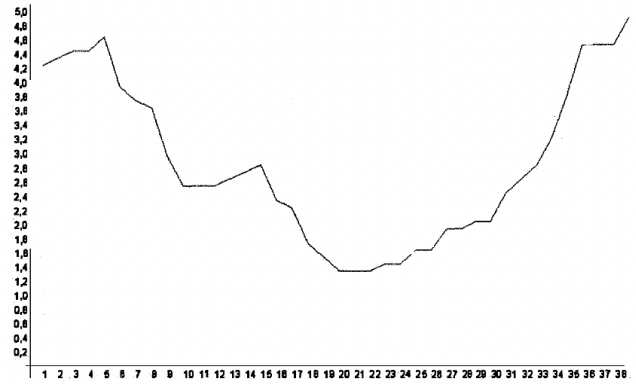
Aufgabe 4

Die Brück & Saar GmbH soll für die AllSun GmbH eine Funktion erstellen, die den Ertrag einer Solaranlage in einem Liniendiagramm (siehe Beispiel perforierte Anlage) darstellt Die Ertragswerte von 365 Tagen sind in der Tabelle Energieertrag gespeichert (siehe nachfolgende Anlage).

a) Erstellen Sie die Funktion MaxErtrag(), die den maximalen Ertragswert aus der Tabelle Energieertrag ermittelt und alle Ertragswerte in dem globalen Array e_werte speichert (Darstellung in Pseudocode, PAP oder Struktogramm). (10 Punkte)

Anlagen:

Beispiel für Liniendiagramm (Ausschnitt)



Tag	Maximaler Ertragswert
1	4,2
2	4,3
3	4,4
...	
365	3,5

Folgender Funktionen wurden bereits erstellt:

Funktion	Beschreibung
Lese EnergieertragSatz():Satz	Liest einen Datensatz aus der Tabelle Energieertrag; Rückgabe ist eine Variable vom Typ Satz. Hinweis: - Lesen erfolgt im richtigen Zeitraum und nach aufsteigendem Datum - Die Datenstruktur Satz entspricht dem Aufbau eines Datensatzes der Tabelle Energieertrag.
zeichneXAchse()	Zeichnet x-Achse mit Beschriftung 1 bis 365
ZeichenYAchse (max_wert:Double)	Zeichnet Y-Achse (Beschriftung: 0 bis maximaler Ertragswert) Hinweis: - Der maximale Energieertragswert muss zuvor ermittelt werden, da er als Parameter übergeben wird. - Die Funktion legt eine geeignete Skalierung der y-Achse fest.
zeichneLinie(tag_1: Int, ertrag_1: Double, tag_2: Int, ertrag_2: Double)	Zeichnet Linie von der Position (tag_1,ertrag_1) bis zur Position (tag_2, ertrag_2)

Lösung

```
Funktion: MaxErtrag()

Legende:  max_wert: Double
          satz: Satz
          e_werte: Array für 365 Ertragswerte
          i: Integer

maxWert = 0
für i = 1 bis 365
    satz = leseEnergieertragsatz()
    e_werte(i) = satz.Energieertrag
    wenn satz.Energieertrag > max_wert
        max_wert = satz.Energieertrag
    ende wenn
ende für

Rückgabe max_wert
```

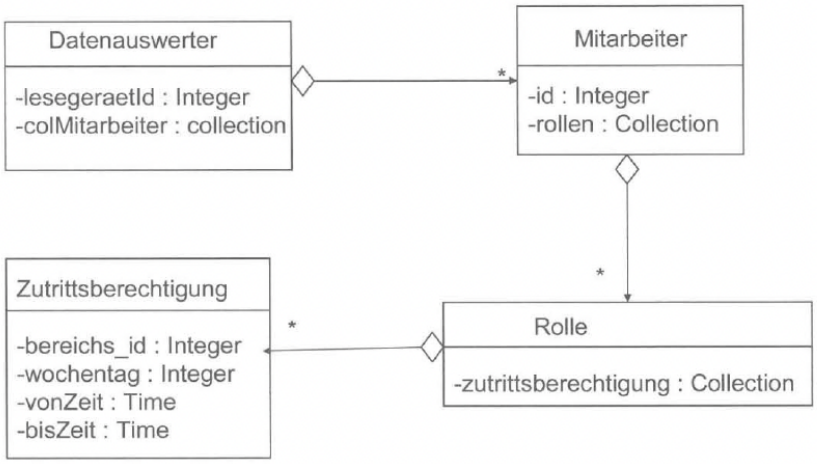
Aufgabe 5

Die XSEC AG hat die Zugangskontrolle wie folgt konzipiert:

Die Mitarbeiter/-innen der Schubert GmbH erhalten Zugangsberechtigungen über ein Rollenkonzept.

Ein Mitarbeiter hat eine oder mehrere Rollen. Für jede Rolle sind entsprechende Zugangsberechtigungen (Bereiche und Zutrittszeiten) festgelegt. Die Bereiche werden mit Lesegeräten versehen, an denen die Mitarbeiter/-innen ihre jeweilige ID eingeben.

Es wurde bereits folgendes Klassendiagramm erstellt.



Jede Collection besitzt u. a. die folgenden Methoden:

Methode	**Beschreibung**
length()	Liefert die Anzahl der Elemente der Collection
get(index : Integer)	Liefert die Objektreferenz des Elementes an der Position index

In jeder Klasse sind für jede Eigenschaft öffentliche get-Methoden vorhanden.

Erstellen Sie auf der Folgeseite für die Klasse *Datenauswerter* eine Methode *zutrittspruefung*, deren Rückgabewerte true oder false sind, je nachdem, ob der Zutritt gewährt wird oder nicht. Die Methode soll mit folgenden Parametern aufgerufen werden:

- MA_ID: Integer
- bereichs_ID: Integer
- wochentag: Integer
- uhrzeit: Time

(19 Punkte)

Lösung

```
Zutrittspruefung(MA_ID: Integer, bereichs_ID: Integer, wochentag: Integer, uhrzeit: Time) :Boolean

zutritt = false
Für i = 0, i < colMitarbeiter.length() oder zutritt = true
    Mitarbeiter mitarbeiter = colMitarbeiter.get(i)
    Wenn mitarbeiter.getId() == MA_ID
        Collection colRolle = mitarbeiter.getRollen()
        Für j = 0, j < colRolle.length() oder zutritt = true
            Rolle rolle = colRolle.get(j)
            Collection colZutritt = rolle.getZutrittsberechtigung();
            Für k = 0, k < colZutritt.length() oder zutritt = true
                Zutrittsberechtigung zb = colZutritt(i)
                Wenn wochentag == zb.getWochentag() und
                        Bereichs_ID =zb.getBereichs_id() und
                        Uhrzeit >= zb.getVonZeit() und uhrzeit <= zb.getBisZeit()
                            zutritt = true
                Ende Wenn
            Ende für
        Ende für
    Ende wenn
Ende für

Rückgabe zutritt
```

Aufgabe 6

Das Mitarbeiterverwaltungssystem speichert folgende Zugangsdaten chronologisch in einer Protokolldatei.

```
Datum;Zeit;Bereichs_ID;Mitarbeiter_ID;Erlaubnis;Zugang/Abgang
12.11.2008;07:45;B22;0798;true;Z
12.11.2008;08:11;B21;0811;true;Z
12.11.2008;08:15;B21;0019;true;Z
12.11.2008;09:46;B21;0902;false;Z
12.11.2008;09:47;B21;1221;true;Z
12.11.2008;11:17;B21;0811;true;A
```

Erstellen Sie die Prozedur *ErmittleMitarbeiterImBereich*(Bereich_ID: Integer), die anhand der Protokolldatei die IDs der Mitarbeiter/-innen in eine Liste ausgibt, die sich in einem bestimmten Bereich aufhalten. Die Bereichs_ID wird der Prozedur als Parameter übergeben.

Zur Lösung dieser Aufgabe können Sie Arrays beliebigen Typs verwenden, ohne eine Dimensionierung vorzugeben. Diese Arrays besitzen stets ausreichend Speicherplatz. (17 Punkte)

Folgende Funktionen stehen Ihnen zur Verfügung:

Funktion	**Beschreibung**
leseProtokollsatz	Liest den nächsten Protokollsatz ein und speichert die durch Semikolon getrennten Informationen in einem Array (6 Stringelemente)
schreibeInArray(Array,ArrayElement)	Speichert das angegebene ArrayElement in das angegebene Array
löscheAusArray(Array,ArrayElement)	Löscht das angegebene ArrayElement aus dem angegebenen Array

Lösung

```
ErmittleMitarbeiterImBereich(Bereich_ID: Integer)

Erstelle MA_ID_Array vom Typ Integer
Erstelle Protokoll_Array vom Typ String

Protokoll_Array = leseProtokollsatz
Solange nicht eof(Protokolldatei)
    Wenn Protokoll_Array[2] = Bereichs_ID und Protokoll_Array[4] = "true" dann
            tempMerker = false
            Für i := 0 bis Länge von MA_ID_Array - 1
                    Wenn Protokoll_Array[3] = MA_ID_Array[i] dann
                            tmpMerker=true
                    Ende wenn
            Ende Für
            Wenn tmpMerker = true dann
                            löscheAusArray(MA_Id_Array, Protokoll_Array[3])
            Sonst
                            schreibeInArray(MA_Id_Array, Protokoll_Array[3])
            Ende wenn
    Ende wenn
    Protokoll_Array = leseProtokollsatz
Ende Solange
Für i := 0 bis Länge von MA_ID_Array - 1
        Ausgabe MA_ID_Array[i]
Ende Für
```

Aufgabe 7

Für den Onlineversand sollen die Weinflaschen mit einem Barcode-Aufkleber versehen werden, der nähere Informationen zum Wein enthält. Der Barcode besteht aus zehn Ziffern.

123-456-78-9-p
- Ziffern 1 bis 3: Schlüssel für Region
- Ziffern 4 bis 6: Schlüssel für Rebsorte
- Ziffern 7 und 8: Jahrgang
- Ziffer 9: Geschmacksangabe (lieblich, halbtrocken, trocken …)
- Dazu kommt an der 10. Stelle eine Prüfziffer p.

Die Entwürfe der Algorithmen sollen als PAP, als Struktogramm oder im Pseudocode erfolgen.

a) Die Prüfziffer soll nach folgender Beschreibung errechnet werden:
   - Die einzelnen Ziffern werden alternierend gewichtet von links nach rechts mit 1 und 3:
     - *Ziffer_1 * 1, Ziffer_2 * 3, Ziffer_3 *1 …Ziffer_9 * 1*
   - Die 9 gewichteten Produkte werden addiert.
   - Die Prüfziffer ist die Differenz der Summe zum nächstkleineren Vielfachen von 10 *(modulo 10)*.

Erstellen Sie auf der Folgeseite eine Funktion „ermittlePrüfziffer", der ein Integerarray mit den 9 Ziffern des Barcodes übergeben wird und die die ermittelte Prüfziffer zurückgibt. 10 Punkte

ermittlePrüfziffer(Integer[ ]) : Integer

Lösung

```
FUNKTION ermittlePrüfziffer( INTEGER[]) : INTEGER
   INTEGER i;
   INTEGER pz = 0;
   INTEGER summe = 0;
   INTEGER produkt = 0;

   VON i=0 BIS 8
      WENN i MODULO 2 == 0 DANN
         produkt = ziffern[i] * 1;
      SONST
         produkt = ziffern[i] * 3;
      ENDE WENN
      summe = summe + produkt;
   ENDE VON
   pz = summe MODULO 10;
   RÜCKGABE pz;
ENDE FUNKTION
```

Aufgabe 8

Alle aktuell vorhandenen Barcodes der Weine und deren Jahresabsatz sind in einer zweidimensionalen Tabelle „Absatz" in folgender Form gespeichert:

Region	Rebsorte	Jahrgang	Geschmacksrichtung	Absatz in Stk.
123	456	78	9	46
333	125	20	4	998
...	...	...		

Erstellen Sie auf der Folgeseite eine Funktion „sucheTopseller", die für ein übergebendes Kriterium (0 = Region; 1 = Rebsorte; 2 = Jahrgang; 3 = Geschmacksrichtung) und einen entsprechenden Vorgabewert (z. B. 123 für eine bestimmte Region) den absatzstärksten Wein ermittelt und den Barcode dieses Weines als Zeichenkette ohne Prüfziffer zurückgibt. Das zweidimensionale Array „Absatz" steht in der Funktion „sucheTopseller" zur Verfügung. Gehen Sie davon aus, dass alle Weine einen unterschiedlichen, positiven Absatz haben. 15 Punkte

Hinweis: Für das Zusammenfügen von Zeichenketten kann der + Operator verwendet werden. Gemischte Ausdrücke vom Typ String und Ganzzahl sind möglich.

Zusammensetzung des Barcodes:
Region-Rebsorte-Jahrgang-Geschmacksrichtung
zB: 123-456-78-9

```
sucheTopseller(kriterium: Integer, vorgabewert: Integer):String
```

Lösung

```
FUNKTION sucheTopseller(kriterium : INTEGER, vorgabewert:
INTEGER):STRING

     INTEGER maxIndex = -1
     INTEGER maxAbsatz = -1
     String barcode = ""

     VON i = 0 BIS Länge von Absatz - 1

          WENN (Absatz[i][kriterium] == vorgabewert und
              Absatz[i][4] > maxAbsatz)
               maxAbsatz = Absatz[i][4]
               maxIndex = i
          ENDE WENN
     ENDE VON

     barcode = barcode + Absatz[maxIndex][0]
     VON i = 1 BIS 3
          barcode = barcode + "-" + Absatz[maxIndex][i]
     ENDE VON

     RÜCKGABE barcode

ENDE FUNKTION
```

Aufgabe 9

Die Verleihfirma möchte ihren Mitarbeitern die Möglichkeit geben, jederzeit eine aktuelle Auswertung ihrer erfassten Arbeitszeiten eines Monats zu erhalten.

Angaben zur Zeiterfassung:
- Für jeden Tag werden maximal zwei Zeiten erfasst, Kommen- und Gehen-Zeit. (Pausen werden nicht berücksichtigt.)

Die Zeiterfassungsliste, die alle Buchungen eines Mitarbeiters für einen Monat anzeigt, soll wie folgt aufgebaut werden (siehe auch Beispiel).
- Liegen für einen Tag die Kommen- und Gehen-Buchungen vor, werden diese Zeiten und die berechnete Anwesenheitszeit in Stunden und Minuten angegeben.
- Liegt für einen Tag nur eine Zeitbuchung vor, ist diese Zeit als Kommen-Zeit, die Anwesenheitszeit 00:00 und der Text „Buchung fehlt" auszugeben.
- Liegt für einen Tag keine Zeitbuchung vor, ist die Anwesenheitszeit 00:00 und der Text „nicht anwesend" auszugeben.
- Zum Ende der Liste ist die Summe der Anwesenheitszeiten auszugeben.

Die Kommen- und Gehen-Zeiten eines Mitarbeiters für einen Monat liegen in einer zweidimensionalen Zeiterfassungstabelle vor.

Beispiel

Zeiterfassungsliste

```
Mitarbeiter: 12345          Oktober 2019

Tag  Kommen  Gehen   Anwesenheit  Bemerkung
================================================
1                    00:00        nicht anwesend
2    08:10   17:20   09:10
3    07:50           00:00        Buchung fehlt
4                    00:00        nicht anwesend
5                    00:00        nicht anwesend
6    08:00   16:00   08:00
7    16:30           00:00        Buchung fehlt
8    08:20   16:40   08:20
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
30   08:10           00:00        Buchung fehlt
31                   00:00        nicht anwesend
================================================
Summe Anwesenheit:  43:10
```

Zeiterfassungstabelle

Tag	Stunde	Minute
2	8	10
2	17	20
3	7	50
6	8	00
6	16	00
7	16	30
8	8	20
8	16	40
~~	~~~	~~~
30	8	10

Erstellen Sie für die Methode ‚erzeugeListe()' einen entsprechenden Algorithmus in Pseudocode, Struktogramm oder PAP.

Folgende Funktionen sind bereits implementiert:

tageImMonat(monat : int, jahr : int) : int	Ermittelt die Anzahl der Tage für den übergebenen Monat eines Jahres.
schreibeKopf(persnr : int, jahr : int, monat : int)	Gibt die Kopfzeilen der Liste aus.
schreibeZeile(tag : int, std1 : int, min1 : int, std2 : int, min2 : int, anwTag : int, bemerkung : String)	Gibt eine Datenzeile aus. Für fehlende Zeiten ist der Wert -1 anzugeben. Die Tagesanwesenheit wird der Funktion in Minuten übergeben und von ihr in Stunden:Minuten ausgegeben.
schreibeFuss(anwMonat : int)	Gibt die Fußzeile aus. Die Monatsanwesenheit wird der Funktion in Minuten übergeben und von ihr in Stunden:Minuten ausgegeben.

Lösung

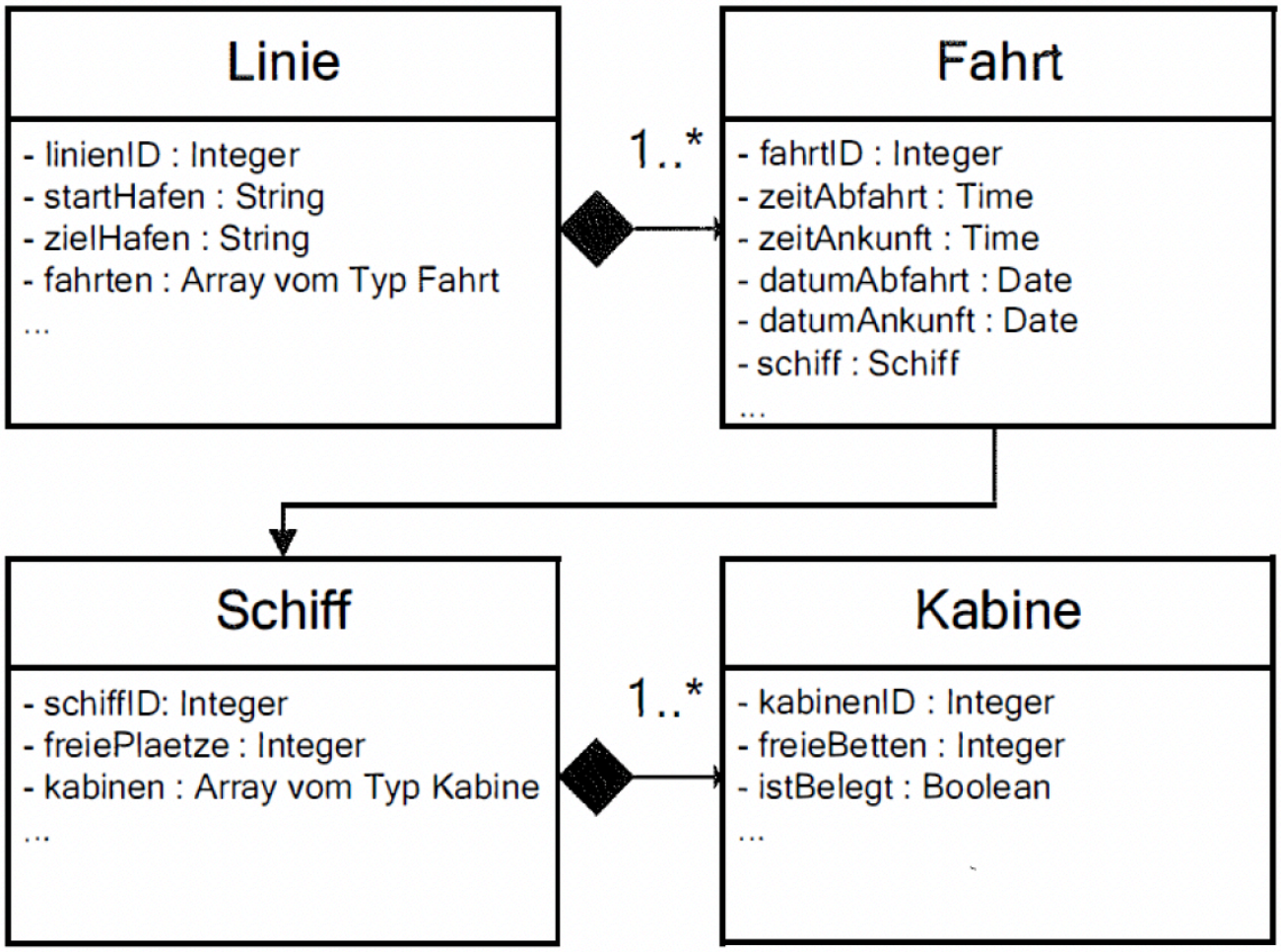
```
erzeugeListe(persnr: int, zeiten: zweidim Tabelle vom Typ int, jahr : int, monat :
int)

// Variablen
anwesenheitTag
anwesenheitMonat := 0
tag    := 1
zeile := 0

schreibeKopf (persnr, jahr, monat)

solange (zeile < Anzahl Zeilen in zeiten)
   anwesenheitTag := 0
   // Keine Buchung
   wenn tag < zeiten[zeile][0]
     dann
       // Keine Buchung
       schreibeZeile(tag, -1, -1, -1, -1, anwesenheitTag," nicht anwesend")
     sonst
       // Zwei Buchungen
       wenn tag = zeiten[zeile][0] und tag = zeiten[zeile + 1][0]
         dann
           anwesenheitTag :=   zeiten[zeile+1][1] * 60 + zeiten[zeile+1][2]
                             - zeiten[zeile][1] * 60 - zeiten[zeile][2]
           anwesenheitMonat := anwesenheitMonat + anwesenheitTag;
           schreibeZeile(tag, zeiten[zeile][1], zeiten[zeile][2],
                         zeiten[zeile+1][1], zeiten[zeile+1][2],
                         anwesenheitTag,
                        "")
           zeile := zeile + 2
         sonst
         // Eine Buchung
           schreibeZeile(tag,
                         zeiten[zeile][1], zeiten[zeile][2],
                         -1, -1
                        anwesenheitTag,
                        " Buchung fehlt");
           zeile := zeile + 1
       endewenn
   endewenn

   tag := tag + 1;

ende solange

// Keine Buchungen am Monatsende, Tageszähler ist kleiner als die Anzahl der
// abgelaufenen Tage im Monat

solange tag <= tageImMonat(monat,jahr)
 schreibeZeile(tag, -1, -1, -1, -1, anwesenheitTag, "nicht anwesend")
 tag := tag + 1
ende solange

// Fusszeile ausgeben
schreibeFusszeile(anwesenheitMonat);
```

Aufgabe 10

Die EASY-Travel GmbH will Online-Buchungen auch für Fährverbindungen einführen.

Die Software für dieses Buchungssystem soll in einer objektorientierten Programmiersprache realisiert werden. Es sind bereits folgende Klassen erstellt worden:

Klassendiagramm



Erläuterungen:

Eine Kabine kann als belegt gekennzeichnet werden, auch wenn nicht alle Betten belegt sind (z. B. eine Doppelbettkabine wird als Einzelbettkabine genutzt).

Für die Eigenschaften der Klassen gibt es öffentliche get-Methoden, z. B.: getStartHafen( ) in der Klasse Linie.

Mit einer Methode der Klasse *Linie* soll geprüft werden, ob an einem bestimmten Tag die vom Kunden gewünschte Anzahl Plätze und Betten auf den Schiffen einer Linie verfügbar sind.

Übergabe- und Ausgabedaten der Methode

Übergabedaten:
- Datum der Abfahrt
- Anzahl der reisenden Personen
- Anzahl der gewünschten Kabinenbetten

Ausgabedaten:
- Fahrt-ID
- Abfahrtsdatum
- Ankunftsdatum
- Abfahrtszeit
- Ankunftszeit
- Kabinenbetten verfügbar Ja/Nein
- Plätze verfügbar Ja/Nein

Beispiel einer Ausgabe

ID028 01.11.0615:00/02.11.0608:30 Plätze nicht verfügbar
ID029 01.11.0622:00/03.11.06 15:30 Plätze verfügbar, Betten nicht verfügbar
Erstellen Sie für die Klasse *Linie* die entsprechende Methode *zeigeVerfügbarkeit( ).*

Hinweis: Stellen Sie die Logik in Code (Pseudocode oder an eine Programmiersprache angelehnten Notation) dar.

Lösung

Pseudocode

```
zeigeVerfügbarkeit(datum: Date, personen: Integer, betten: Integer)

für i = 0 bis Anzahl fahrten – 1

        wenn fahrten[i].getDatum() = datum dann
            Ausgabe = fahrten[i].getFahrtId(),
                      fahrten[i].getDatumAbfahrt(), fahrten[i].getZeitAbfahrt(),
                      fahrten[i].getDatumAnkunft(), fahrten[i].getZeitAnkunft()

            schiff = fahrten[i].getSchiff()
            wenn schiff.getFreiePlaetze() >= personen dann
              Ausgabe "Plätze vorhanden"
              wenn betten > 0 dann
                freieBetten = 0
                kabinen = schiff.getKabinen()
                für j = 0 bis Anzahl kabinen – 1
                  wenn nicht kabinen[j].istBelegt() dann
                    freieBetten = freieBetten + kabinen[j].getFreieBetten()
                  ende wenn
                ende für
              wenn freieBetten >= betten dann
                Ausgabe "Betten verfügbar"
              sonst
                Ausgabe "Betten nicht verfügbar"
              ende wenn
            ende wenn
        sonst
            Ausgabe "Plätze nicht verfügbar"
        ende wenn
    ende wenn
ende für
```

Aufgabe 11

b) Eine Methode Ausgabe soll eine Ergebnisliste erstellen, die alle Sportler eines Wettkampfs und deren jeweiliges Ergebnis enthält. Der Methode wird eine Referenzvariable auf ein Wettkampfobjekt übergeben.

Diese Methode soll die folgende Bildschirmausgabe ermöglichen:
Ergebnis-Liste: Disziplin: 100-Meter-Lauf/Wettkampf: Endlauf
Lothar Hermes 8,9
Armin Hurry 10,3
Speedi Conzales 7,3

Entwickeln Sie diese Methode. Verwenden Sie hierfür Pseudocode. (8 Punkte)

Lösung

```
Ausgabe(„Ergebnisliste-Disziplin:")
Disziplin disz = wk.getDisziplin()
Ausgabe(disz.getName())
Ausgabe(„/");
Ausgabe(wk.getName())
Für i = 1 bis Länge von Array sportler
    Ausgabe(sportler[i].getName())
    Ausgabe(wert[i])
```

Aufgabe 12

Die bildgebende Diagnostik liefert täglich viele Dateien, die gespeichert werden müssen. Um Speicherplatz einzusparen, soll ein Komprimierungsalgorithmus entwickelt werden. Für einen ersten Prototypen wurde folgende Vorgabe erstellt.

Vorgabe:
Die Bilddaten sollen mit einer Lauflängenkodierung komprimiert werden. Dabei werden sich direkt wiederholende Zeichen zusammengefasst, und nur die Anzahl und das entsprechende Zeichen erfasst. Eine Zusammenfassung soll erst bei mehr als vier Zeichen erfolgen. Zur Erkennung der Lauflängenkodierung wird das „%"-Zeichen verwendet, das in den unkomprimierten Bilddaten nicht vorkommt. Die Bilddaten liegen als String-Arrays vor.

Beispiel:
String[ ] „unkomprimiert"

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]
Z	Z	Z	Z	7	7	7	7	7	7	7	7	7	7	M	P	P	P	P	P	H	H

String[ ] „komprimiert"

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
Z	Z	Z	Z	%	10	7	M	%	5	P	H	H

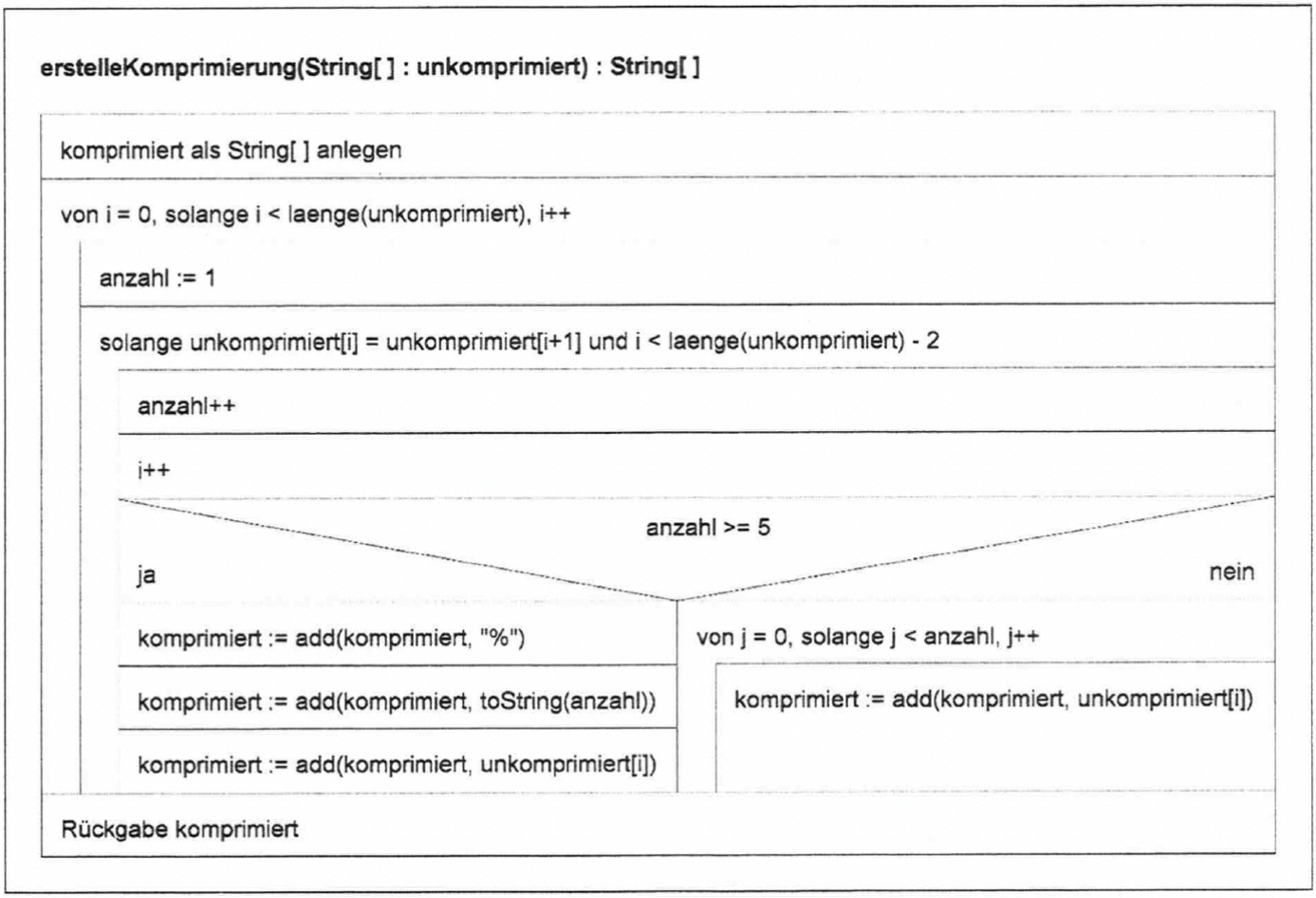
ZZZZ7777777777MPPPPPHH (unkomprimiert)
ZZZZ%107M%5PHH (komprimiert)

Folgende Funktionen stehen zu Verfügung:

laenge(String[ ]) : Integer	Gibt die Länge des übergebenen Zeichenkettenarrays als ganze Zahl zurück
add(String[], String) : String[ ]	Verlängert das übergebene Zeichenkettenarray um den übergebenen String und gibt es zurück
toString(Integer) : String	Gibt die übergebene ganze Zahl als String zurück

a) Entwickeln Sie auf der gegenüberliegenden Seite nach Vorgabe als Struktogramm, PAP oder Pseudocode eine Funktion „erstelleKomprimierung", die aus einem übergebenen unkomprimierten String-Array ein komprimiertes String-Array erstellt und zurückgibt. 20 Punkte

Lösung


erstelleKomprimierung(String[ ] : unkomprimiert) : String[ ]
komprimiert als String[ ] anlegen
von i = 0, solange i < laenge(unkomprimiert), i++
anzahl := 1
solange unkomprimiert[i] = unkomprimiert[i+1] und i < laenge(unkomprimiert) - 2
anzahl++
i++
anzahl >= 5
ja
nein
komprimiert := add(komprimiert, "%")
komprimiert := add(komprimiert, toString(anzahl))
komprimiert := add(komprimiert, unkomprimiert[i])
von j = 0, solange j < anzahl, j++
komprimiert := add(komprimiert, unkomprimiert[i])
Rückgabe komprimiert