Kapitel 6 Lehrermaterial

Lernziele:

Festigung der Konzepte Deklaration, Initialisierung, Felder, Wiederholung mit fester Anzahl und deren Umsetzung mit Java

Einsatz:

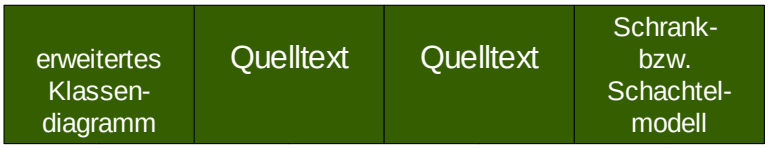
Nach der Bearbeitung von Kapitel 6 durch die Schüler

Unterrichtsform:

Fragend entwickelnder Unterricht

Medien:

Tafel



Aufgabe

Erstellen einer Klasse mit deren Hilfe 24 Temperaturwerte, zu jeder Stunde des Tages einer, gespeichert und verwaltet werden können.

a) Diskussion Attribute

- mehrere Attribute versus Datenfeld
- zwei Datenfelder oder Interpretation des Index als Stundenangabe
- Datentyp String (wegen dem Zeichen °) oder int (Berechnungen wie die Durchschnittstemperatur sind möglich)

Ergebnis (Deklaration)

```
int[] temperaturen
```

Im Schrankmodell wird der Name und die Schubladengröße festgelegt.

b) Diskussion Konstruktor

- Erzeugung des Datenfeldes
  Im Schrankmodell wird die Anzahl der Schubladen festgelegt.
- Initialisierung des Referenzattributs
- Initialisierung der Feldelemente mit Hilfe einer Wiederholung mit fester Anzahl
  Im Schrankmodell wird in jede Schublade ein Zettel mit einem int-Wert gelegt.
- Was ist ein sinnvoller Startwert, der eindeutig kennzeichnet, dass noch kein realer Temperaturwert eingegeben wurde (Analog zu der Zeichnkette "---" bei dem Datenfeld namen der Klasse VERLAUFSLISTE)
  z.B. -300, da unterhalb des absoluten Nullpunkts

Ergebnis

```
1  class TAGESTEMPERATUR
2  {
3      //Referenzattribute
4      int[] temperaturen;
5
6      /**
7       * Konstruktor für Objekte der Klasse TAGESTEMPERATUR
8       */
9      TAGESTEMPERATUR()
```



```
10      {
11          temperaturen = new int[24];
12          for(int zaehler=0; zaehler <= 23; zaehler = zaehler+1)
13          {
14              temperaturen[zaehler] = -300;
15          }
16      }
...



}
```

Abbildung 1: Auszug aus dem Quelltext der Klasse TAGESTEMPERATUR

c) Programmablauf beim Erzeugen eines Objekts

Methode:
Ähnlich wie in einem Debug Modus wird der Programmablauf Zeile für Zeile durchgegangen (jeweils ein farbiger Pfeil mit Kreide kennzeichnet die aktuelle Zeile).

Spätestens hier (das meiste kann acuh schon in b) gemacht werden) sollte der Schrank auch an die Tafel gezeichnet werden.

Funktionsweise der for-Anweisung:

- 1. Durchlauf
  int zaehler=0
  Deklaration und Initialisierung eines (lokalen) Attributs
  Schachtelmodell: An der Tafel wird eine int-Schachtel mit dem Bezeichner zaehler benötigt. Der Zettel in der Schachtel trägt zunächst den Wert 0.
- Beginn der Wiederholungsanweisung:
  zahler <= 23
  Auswertung eines boolschen Ausdrucks mit true --> Die Anweisung(en) im Anweisungsteil der Wiederholungsanweisung wird (werden) ausgeführt.
  Schachtelmodell: Es wird geschaut, welcher Wert der Zettel in der Schachtel "zaehler" hat: 0. Dieser Wert ist kleiner gleich 23 --> true
- Anweisungsteil der Wiederholungsanweisung:
  temperaturen[0] = -300;
  In die Schublade (des Schranks temperaturen) mit der Nummer 0 kommt ein Zettel mit dem Wert -300.
- Ende der Wiederholungsanweisung:
  zaehler = zaehler + 1;
  Aus der Schachtel "zaehler" wird der Zettel genommen, der Wert genommen, um eins erhöht, auf einen neuen Zettel geschrieben. Der alte Zettel wird weggeschmissen, der neue, mit dem Wert 1, in die Schachtel gelegt.

- 2. Durchlauf
  int zaehler=0 wird nicht mehr beachtet
- Beginn der Wiederholungsanweisung:
  zahler <= 23
  Auswertung eines boolschen Ausdrucks mit true --> Die Anweisung(en) im Anweisungsteil der Wiederholungsanweisung wird (werden) ausgeführt.
  Schachtelmodell: Es wird geschaut, welcher Wert der Zettel in der Schachtel "zaehler" hat: 1. Dieser Wert ist kleiner gleich 23 --> true
- Anweisungsteil der Wiederholungsanweisung:
  temperaturen[1] = -300;
  In die Schublade (des Schranks temperaturen) mit der Nummer 1 kommt ein Zettel mit dem Wert -300.
- Ende der Wiederholungsanweisung:
  zaehler = zaehler + 1;
  Aus der Schachtel "zaehler" wird der Zettel genommen, der Wert genommen, um eins erhöht , auf

einen neuen Zettel geschrieben. Der alte Zettel wird weggeschmissen, der neue, mit dem Wert 2, in die Schachtel gelegt.

...

24. Durchlauf
int zaehler=0 wird nicht mehr beachtet
- Beginn der Wiederholungsanweisung:
  zahler <= 23
  Auswertung eines boolschen Ausdrucks mit true --> Die Anweisung(en) im Anweisungsteil der Wiederholungsanweisung wird (werden) ausgeführt.
  Schachtelmodell: Es wird geschaut, welcher Wert der Zettel in der Schachtel "zaehler" hat: 23. Dieser Wert ist kleiner gleich 23 --> true
- Anweisungsteil der Wiederholungsanweisung:
  temperaturen[23] = -300;
  In die Schublade (des Schranks temperaturen) mit der Nummer 23 kommt ein Zettel mit dem Wert -300.
- Ende der Wiederholungsanweisung:
  zaehler = zaehler + 1;
  Aus der Schachtel "zaehler" wird der Zettel genommen, der Wert genommen, um eins erhöht , auf einen neuen Zettel geschrieben. Der alte Zettel wird weggeschmissen, der neue, mit dem Wert 24, in die Schachtel gelegt.

25. Durchlauf
int zaehler=0 wird nicht mehr beachtet
- Beginn der Wiederholungsanweisung:
  zahler <= 23
  Auswertung eines boolschen Ausdrucks mit false --> Die Wiederholungsanweisung wird beendet.
  Schachtelmodell: Es wird geschaut, welcher Wert der Zettel in der Schachtel "zaehler" hat: 24. Dieser Wert ist NICHT kleiner gleich 23 --> false

d) Diskussion einer Methode zum Eingeben von Temperaturwerten

- Methodenkopf: Eingabeparameter / Ausgabewerte
- Methodenrumpf

Ergebnis

```
17
18      void TemperaturEingeben(int temperaturNeu, int stunde)
19      {
20          temperaturen[stunde]= temperaturNeu;
21      }
```

Abbildung 2: Quelltext der Methode *TemperaturEingeben*

e) Programmablauf beim Methodenaufruf

Methode:
Debug- Modus (siehe oben). Dieser Ablauf ist sehr wichtig, da er den Unterschied zwischen Methode (allgemeine Fähigkeit) und Methodenaufruf (konkrete Handlungsanweisung) verdeutlicht.

- Zeile 18: In BlueJ geht ein Fenster auf, in das man konkrete Werte für die Parameter temperaturNeu und stunde eingeben muss, z.B. 4 und 6.
  Eventuell zwei neue Schachteln für die Parameter anzeichnen.
- Zeile 20: temperaturen[6] = 4;
  Im Schrankmodell wird der Zettel in der Schublade 6 durch einen Zettel mit dem Wert 4 ersetzt.

f) Optimierung der Methode TemperaturEingeben

Diskussion, was bei dem Methodenaufruf
TemperaturEingeben(10, 50)
passiert und wie man das ungewollte verhindern kann.

Ergebnis

```
void TemperaturEingebenOptimiert(int temperaturNeu, int stunde)
    {
        if(stunde >= 0 && stunde <=23)
        {
            temperaturen[stunde]= temperaturNeu;
        }
        else
        {
            System.out.println("Der Wert fuer die Stunde darf weder
                    kleiner als 0 noch groesser als 23 sein.");
        }
    }
```

Hinweise:
- Der logische Operator in Java && muss den Schülern genannt werden, sie sollen die Bedingung aber selbst verbal formulieren. Die Funktionsweise des Operators ist aus Jgst. 9 bekannt.
- Der else-Zweig ist optional.

g) Weitere Möglichkeiten im Kontext dieses Beispiels

siehe Quelltext.

```
/**
 * In der Klasse TAGESTEMPERATUR koennen vierundzwanzig Temperaturwerte
 * (in Grad Celsius) gespeichert werden. Ueber den Index des Datenfeldes
 * temperaturen sind sie einer Stunde zugeordnet, z.B. ist der Wert des
 * Feldelements temperaturen[10] die Temperatur um 10 Uhr.
 *
 * @author (Peter Brichzin)
 * @version (1.12.08)
 */
class TAGESTEMPERATUR
{
    //Referenzattribute
    int[] temperaturen;

    /**
     * Konstruktor für Objekte der Klasse TAGESTEMPERATUR
     */
    TAGESTEMPERATUR()
    {
        temperaturen = new int[24];
        for(int zaehler=0; zaehler <= 23; zaehler = zaehler+1)
        {
            temperaturen[zaehler] = -300;
        }
    }

    /**
     * Die Methode TemperaturwertEingeben ermoeglicht die Eingabe
     * einer Temperatur zugeordnet zu einer Stunde.
```



```
     *
     * @param  temperaturNeu    neuer Temperaturwert in Celsius
     * @param  stunde           Stunde in der die Temperatur gemessen wurde
     */
    void TemperaturEingeben(int temperaturNeu, int stunde)
    {
        temperaturen[stunde]= temperaturNeu;
    }

    /**
     * Die Methode TemperaturwertEingebenOptimiert ermoeglicht die Eingabe
     * einer Temperatur zugeordnet zu einer Stunde. Es wird darauf geachtet,
     * dass der Eingabewert stunde zwischen 0 und 24 liegt.
     *
     * @param  temperaturNeu    neuer Temperaturwert in Celsius
     * @param  stunde           Stunde in der die Temperatur gemessen wurde
     */
    void TemperaturEingebenOptimiert(int temperaturNeu, int stunde)
    {
        if(stunde >= 0 && stunde <=23)
        {
            temperaturen[stunde]= temperaturNeu;
        }
        else
        {
            System.out.println("Der Wert fuer die Stunde darf weder kleiner als 0
noch groesser als 23 sein.");
        }
    }

    /**
     * Die Methode FehlendeTemperaturwerteAnzeigen gibt auf der Konsole
     * Die Stunden als Ganzzahl aus, zu denen noch keine Temperatur eingegeben
     * wurde, d.h. bei denen der Wert -300 als Temperatur gespeichert ist.
     */
    void FehlendeTemperaturwerteAnzeigen()
    {
        for(int zaehler=0; zaehler <= 23; zaehler = zaehler+1)
        {
            if(temperaturen[zaehler] == -300)
            {
                System.out.println("Der Temperaturwert zur folgenden Stunde wurde
noch nicht eingegeben:");
                System.out.println(zaehler);
            };
        }

    }


    /**
     * Die Methode DurchschnittsTemperaturBerechnen berechnet die
     * Tagesdurchschnittsstemperatur und gibt diese auf der Konsole aus.
     * Ein Rueckgabewert wird nicht verwendet, da die Schueler dieses
     * Konzept noch nicht kennen.
     *
     *
     */
    void DurchschnittsTemperaturBerechnen()
    {
        // lokales Attribut
        int durchschnittstemperatur;  // Deklaration
        durchschnittstemperatur=0;    // Initialisierung

        for(int zaehler=0; zaehler <= 23; zaehler = zaehler+1)
        {
            durchschnittstemperatur = durchschnittstemperatur +
temperaturen[zaehler];
        }
        durchschnittstemperatur= durchschnittstemperatur/24;
```



```
        System.out.println("Die Durchscnittstemperatur betraegt:");
        System.out.println(durchschnittstemperatur);


    }

}
```