Kapitel 6 Punkteliste

Lernziele:

Deklaration, Initialisierung, Felder, Wiederholung mit fester Anzahl

6.0 Deklaration und Initialisierung

Kapitel 6 lässt sich mit den Begriffen Deklaration und Initialisierung prägnanter und damit verständlicher erklären. Aus diesem Grunde findet sich hier ein kleiner Einschub, bevor es mit dem Erstellen einer Punkteliste thematisch zu Krümel & Monster losgeht.

In Abbildung 1 ist ein Auszug der Klasse MAMPFI zu sehen, wobei jeder Quelltextzeile eine Zeilennummer zugeordnet ist.

```
1   class MAMPFI
2   {
3       // Attribute
4       int positionX;
5       int positionY;
6       boolean verwundbar;
7       char blickrichtung;
8
9       // Referenzattribut
10      KREISFORM symbol;
11
12      /**
13       * Konstruktor für Objekte der Klasse MAMPFI
14       */
15      MAMPFI()
16      {
17          verwundbar = true;
18          blickrichtung = 'N';
19
20          symbol = new KREISFORM();
21          symbol.RadiusSetzen(50);
22          symbol.StartWinkelSetzen(120);
23          symbol.BogenWinkelSetzen(300);
24          symbol.BogenArtSetzen(2);
25          symbol.FarbeSetzen("gelb");
26      }
        //weitere Methoden
    }
```

Abbildung 1: Auszug aus dem Quelltext der Klasse MAMPFI

In Zeile 6 wird festgelegt, dass es ein Attribut vom Typ boolean mit dem Namen verwundbar gibt. Das Vereinbaren des Namens und des Datentyps für ein Attribut nennt man (Attribut-)**Deklaration**[1]**.**

Aufgabe 6.1
In welchen Zeilen der Abbildung 1 finden noch Deklarationen statt?

Aufgabe 6.2
In Kapitel 3 wurde auf das Schachtelmodell verwiesen: Jedes Attribut kann man sich als eine Schachtel mit folgenden Eigenschaften vorstellen:
- Entsprechend den Datentypen gibt es in der Größe genormte Schachteln.
- Jede Schachtel wird mit dem Attributbezeichner beschriftet.
- In der Schachtel liegt ein Zettel mit dem aktuellen Attributwert.

Welche(r) dieser drei Punkte werden durch die Deklaration festgelegt? Begründe deine Antwort.

1 engl. declaration: Vereinbarung, Anmeldung, Bekanntmachung

Jedem Attribut muss ein sinnvoller Anfangswert zugewiesen werden. Diesen Schritt nennt man **Initialisierung**[2]. In Java finden die Initialisierungen im Konstruktor statt. Beispielsweise ist in der Klasse MAMPFI (Abbildung 1) die Initialisierung des Attributs verwundbar in Zeile 17 zu finden.

Aufgabe 6.3
a) In welchen Zeilen der Abbildung 1 finden noch Initialisierungen statt?
b) Das Objekt der Klasse KREISFORM, welches durch das Referenzattribut symbol referenziert wird hat auch Attribute. Wo werden diese initialisiert?

Aufgabe 6.4
Welcher Vorgang im Schachtelmodell entspricht der Initialisierung (im Quelltext)?

2 engl. init: Anfang

6.1 Punkteliste der letzten bzw. besten Spieler

In vielen Spielen wird auf Knopfdruck eine Liste mit den Namen der letzten Spieler und ihren Punkten (Verlauf, engl: history) angezeigt (Abbildung 2). Viel häufiger werden nicht die letzten, sondern die besten Spieler und die von ihnen erzielten Punkte angezeigt (engl.: Highscore). Da die Bestenliste ein gutes Stück schwieriger als die Verlaufsliste ist, behandelt das folgenden Kapitel das Erstellen einer Verlaufsliste. Wer dann noch die schwierige Variante bearbeiten möchte, hat die Möglichkeit dies in den Aufgaben zu tun, die mit dem  Symbol gekennzeichnet sind.

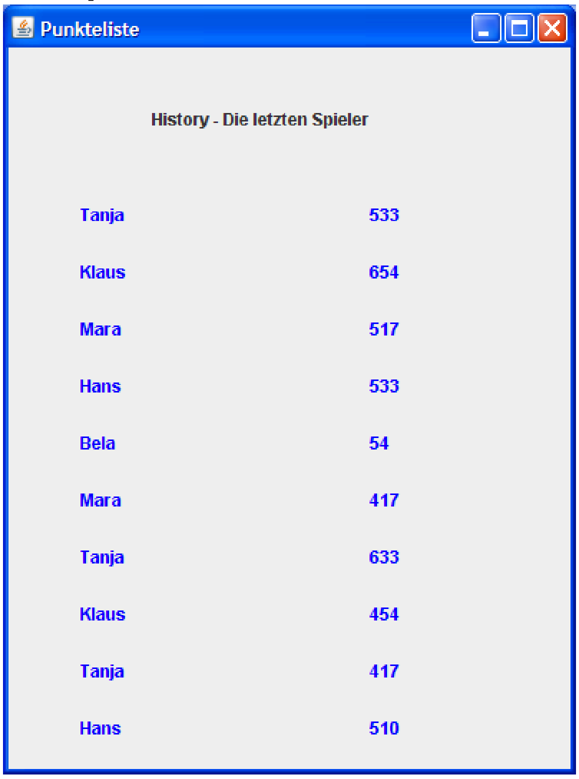
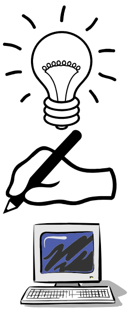
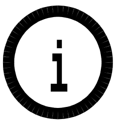


Abbildung 2: Verlaufsliste



Aufgabe 6.5
Modelliere eine Klasse VERLAUFSLISTE und setze sie in Java um. Beschränke dich auf die wesentlichen Attribute und Methoden. In einem ersten Schritt sind drei Namenseinträge mit den zugehörigen Punkten ausreichend.



Vorgehensweise

Hinweise zur Vorgehensweise:

1) In Kapitel 3 und 4 wurde in folgenden Schritten vorgegangen (siehe auch Abbildung 3):
   i) Erstellen eines Klassendiagramms
   ii) Erstellen eines erweiterten Klassendiagramms
   iii) Übertragen des Klassendiagramms in Java-Quelltext
   Gehe sowohl hier in der Aufgabe als auch in Zukunft derart schrittweise vor.

2) Beginne das Implementieren, indem du in BlueJ ein neues Projekt mit dem Namen kap06 erzeugst.
   (Es ist generell sinnvoll, die Ergebnisse jeden Kapitels unter einem eigenen Namen abzuspeichern. So kann man jederzeit auf lauffähige Versionen früherer Kapitel zurückgreifen.)

3) Versuche dich an dieser Aufgabe zunächst alleine bzw. mit deinem Nachbar. Vergleiche DANACH mit den Lösungen auf den beiden folgenden Seiten. So hast du eine gute Kontrolle, ob deine Lösungsansätze passen.

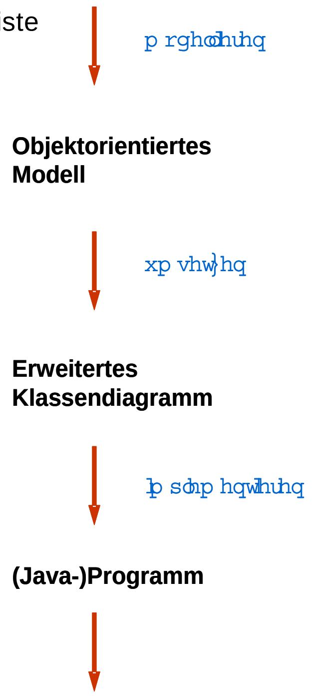


Abbildung 3: Vorgehensweise beim Modellieren und Programmieren

Die Klasse VERLAUFSLISTE benötigt drei Attribute zum Speichern von Namen und drei Attribute zum Speichern von Punkten (Abbildung 4). Für die Namen eignet sich als Datentyp eine Zeichenkette (String) und für die Punkte eine ganze Zahl (int) (Abbildung 5). Neben dem Konstruktor muss es eine Methode für den Eintrag eines neuen Ergebnisses geben. Als Eingabewerte der Methode *NeuesErgebnisEintragen* sind ein Name und eine Punktzahl nötig (Abbildung 5).

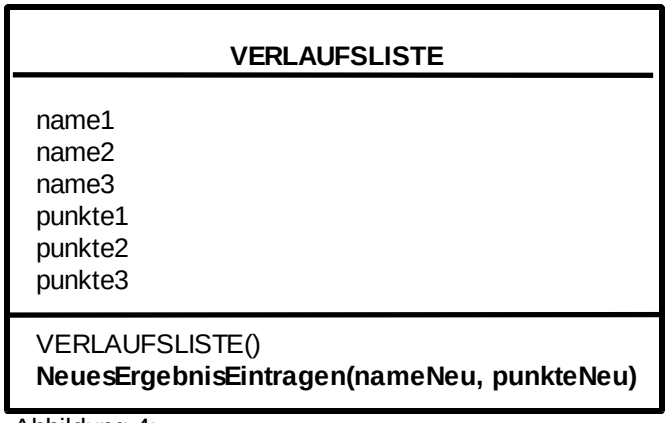


Abbildung 4:
Klassendiagramm der Klasse VERLAUFSLISTE

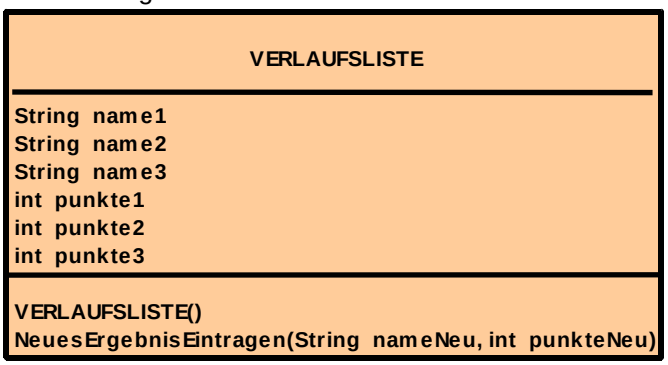


Abbildung 5: erweitertes Klassendiagramm der Klasse VERLAUFSLISTE

Hinweis:
Beachte, dass der Abschnitt der Attribute im erweiterten Klassendiagramm (Abbildung 5) die (Attribut-)Deklaration ist.

Aufgabe 6.6
Welche Attributwerte sind bei der Erzeugung eines Objekts der Klasse VERLAUFSLISTE sinnvoll? Wie sieht konkret die Initialisierung im Konstruktor aus?

Der Konstruktor hat die Aufgabe ein Objekt bei der Erzeugung in einen sinnvollen Anfangszustand zu versetzen. Beim Erzeugen sollte die Verlaufsliste leer sein, d.h. sie sollte keine Namen enthalten und die Punkte sollten auf 0 gesetzt sein. Somit sind sechs Zuweisungen im Konstruktor nötig, um alle Attribute zu initialisieren (Abbildung 6).

Hinweis:
Als Ausdruck dafür, dass kein Name eingetragen ist werden hier drei Bindestriche verwendet.

Rumpf des Konstruktors VERLAUFSLISTE

```
name1 = "---";
name2 = "---";
name3 = "---";
punkte1 = 0;
punkte2 = 0;
punkte3 = 0;
```

Abbildung 6:
Struktogramm zur Planung des Konstruktors

Aufgabe 6.7
Wie verändert sich die Verlaufsliste beim Eintragen eines neuen Spielergebnisses? Was passiert z.B. mit dem ersten Eintrag, was mit dem letzten Eintrag? Wie sieht der Rumpf der Methode *NeuesErgebnisEintragen* aus?
Setze die Methode *NeuesErgebnisEintragen* in deiner Klasse VERLAUFSLISTE um.

Beim Eintrag eines neuen Spielergebnisses wird in unserem Fall der letzte (hier dritte) Eintrag gelöscht, der zweite auf den letzten "verschoben", der erste auf den zweiten verschoben und die neuen Spielergebnisse im ersten Eintrag gespeichert. Ein expliziter Löschvorgang des dritten Eintrags ist sogar nicht nötig, weil man ihn mit dem zweiten Eintrag der Liste überschreibt. Abbildung 7 zeigt das Struktogramm der Methode *NeuesErgebnisEintragen.*

Rumpf der Methode NeuesErgebnisEintragen(nameNeu, punkteNeu)

```
name3 = name2;
name2 = name1;
name1 = nameNeu;
punkte3 = punkte2;
punkte2 = punkte1;
punkte1 = punkteNeu;
```

Abbildung 7: Struktogramm zur Planung der Methode *NeuesErgebnisEintragen*

Aufgabe 6.8
Die Reihenfolge der Zuweisungen ist sehr wichtig, wie das Beispiel der Methode *NeuesErgebnisEintragenFalsch* (Abbildung 8) zeigt.

Rumpf der Methode NeuesErgebnisEintragenFalsch(nameNeu, punkteNeu)

```
name1 = nameNeu;
name2 = name1;
name3 = name2;
punkte1 = punkteNeu;
punkte2 = punkte1;
punkte3 = punkte2;
```

Abbildung 8: Struktogramm zur Planung der Methode *NeuesErgebnisEintragenFalsch*

Erkläre knapp, warum die Methode *NeuesErgebnisEintragenFalsch* nicht richtig arbeitet. Als Hilfe kannst du einen konkreten Methodenaufruf mit den Eingabewerten *"Emil"* und 333 bei dem Objekt verlaufsliste "durchspielen", dessen Zustand vor dem Aufruf in Abbildung 9 zu sehen ist.
Welche Attributwerte hat dieses Objekt nach dem Aufruf? Gehe die Anweisungen im Methodenrumpf aus Abbildung 8 Zeile für Zeile durch und aktualisiere nach jedem Schritt die Attributwerte des Objekts (z.B. in deinem Heft nachdem du dorthin Abbildung 9 übertragen hast).

verlaufsliste: VERLAUFSLISTE

name1 = " Mara"
name2 = "Tanja"
name3 = "Bela"
punkte1 = 345
punkte2 = 467
punkte3 = 412

Abbildung 9: Zustand des Objekts verlaufsliste voher

Aufgabe 6.9
Erstelle eine Klasse BESTENLISTE, bei der im Gegensatz zur Verlaufsliste die Einträge nach Punkten sortiert sind. Beschränke dich zunächst auch auf 3 Einträge.

6.2 Arbeitsersparnis bei längeren Listen durch Felder

Drei Einträge sind für eine Verlaufs- bzw. Bestenliste sehr wenig. Typisch enthalten die Listen zehn, oft sogar mehr Einträge. Bei zehn Einträgen würde dies bedeuten, dass 20 Attribute deklariert und initialisiert werden müssen. Diese 40 Zeilen zu schreiben ist mit einem gewissen Zeitaufwand verbunden und nicht gerade eine spannende Arbeit. Die Klasse VERLAUFSLISTE sähe aus wie in Abbildung 10.

```
class VERLAUFSLISTE
{
   // Attribute
   String name1;
   String name2;
   String name3;
   String name4;
   String name5;
   String name6;
   String name7;
   String name8;
   String name9;
   String name10;
   int punkte1;
   int punkte2;
   int punkte3;
   int punkte4;
   int punkte5;
   int punkte6;
   int punkte7;
   int punkte8;
   int punkte9;
   int punkte10;

   // Konstruktor
  VERLAUFSLISTE()
   {
        name1 = "---";
        name2 = "---";
        name3 = "---";
        name4 = "---";
        name5 = "---";
        name6 = "---";
        name7 = "---";
        name8 = "---";
        name9 = "---";
        name10 = "---";
        punkte1 = 0;
        punkte2 = 0;
        punkte3 = 0;
        punkte4 = 0;
        punkte5 = 0;
        punkte6 = 0;
        punkte7 = 0;
        punkte8 = 0;
        punkte9 = 0;
        punkte10 = 0;
    }

 // Methoden

}
```

Deklaration

Initialisierung

Abbildung 10: Aufwändiger Quelltextauszug für eine Klasse VERLAUFSLISTE

Um eine solche aufwändige Arbeit zu vermeiden, gibt es in der Programmierung das Konzept des **Datenfeldes**, kurz **Feld** (engl. **array**) genannt. Es ist eine Datenstruktur, in der mehrere Werte des gleichen Datentyps zusammengefasst werden. In unserem Fall muss ein Objekt der Klasse **FELD** zehn Namen verwalten, die alle vom Typ String, d.h. Zeichenketten sind.

Die einzelnen Feldelemente sind durchnummeriert. Über diese Nummer (auch **Index** genannt) kann man auf jedes **Feldelement** zugreifen. Vorstellen kann man sich ein Feld wie einen Aktenschrank mit nummerierten Schubladen. In jeder Schublade hat genau eine Informationseinheit (z. B. eine Zahl oder eine Zeichenkette) Platz (Abbildung 11). Jedoch gibt es pro Schrank nur eine Sorte von Schubladen! Somit gibt es unterschiedliche Schranktypen: Schränke, in deren Schubladen nur ganze Zahlen (Attribute des Datentyps int) passen Schränke, die nur für Attribute des Typs String gemacht sind u.s.w. (Abbildung 11 bzw. 12).
Selbstverständlich muss nicht jede Schublade gefüllt sein.

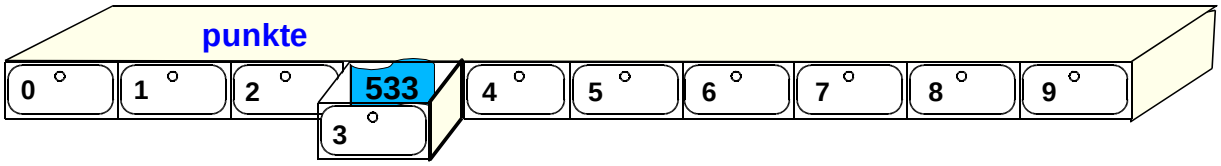


Abbildung 11:
Das Feld punkte im Schrankmodell: In jede der durchnummerierten Schubladen passt genau ein Zettel mit einer ganzen Zahl (int)

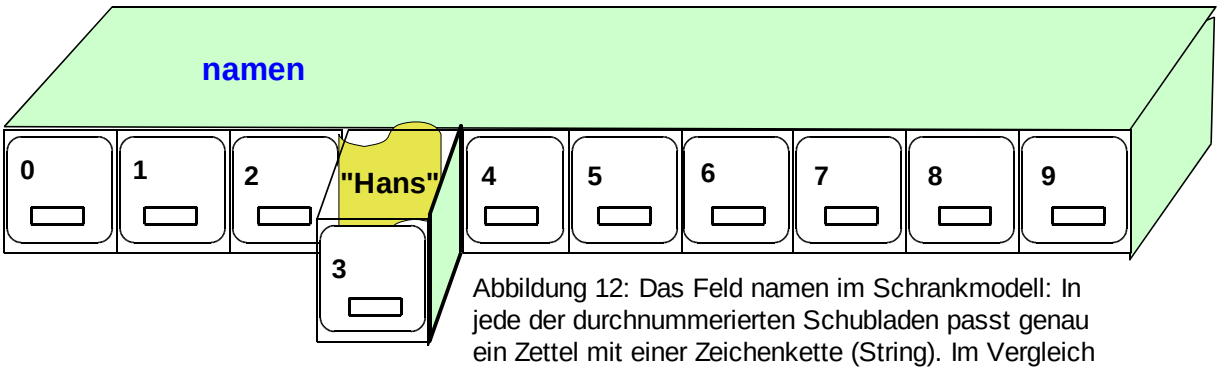


Abbildung 12: Das Feld namen im Schrankmodell: In jede der durchnummerierten Schubladen passt genau ein Zettel mit einer Zeichenkette (String). Im Vergleich zur Abbildung 11 sind die Schubladen größer, weil Zeichenketten einen größeren Speicherbedarf als ganze Zahlen haben.

Insgesamt werden zur Umsetzung einer Verlaufsliste zwei Felder benötigt, eines zum Verwalten der Namen, das andere zum Verwalten der Punkte. In der Abbildung 1 ist eine Verlaufsliste mit jeweils zehn Einträgen gezeigt. Abbildung 13 zeigt das passende Objektdiagramm dazu.

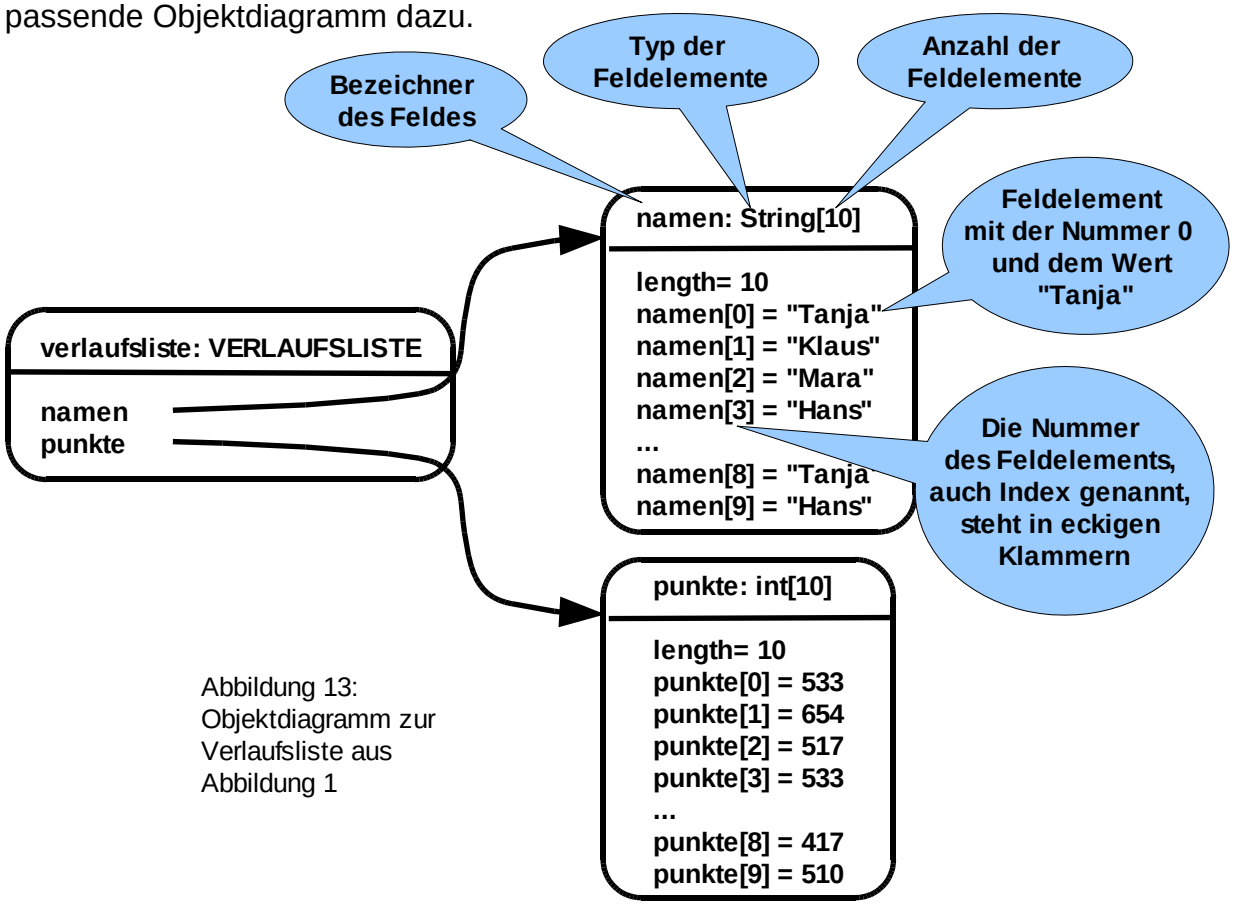
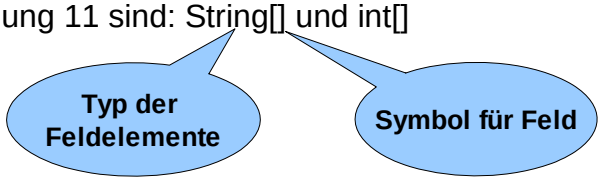


Abbildung 13:
Objektdiagramm zur Verlaufsliste aus Abbildung 1

Hinweise:

- Als Symbol für Felder wird ein Paar eckiger Klammern verwendet. Da Felder nur Elemente eines Typs verwalten können, muss der entsprechende Datentyp immer mit genannt werden. Die entsprechende Kurzschreibweise für die Felder aus Abbildung 11 sind: String[] und int[]



- Die Nummerierung der Feldelemente beginnt immer bei 0, d.h. die Elemente eines Feldes der Länge 10 haben die Nummern 0 bis 9.

In Abbildung 13 wird deutlich, dass zwischen dem Objekt der Klasse VERLAUFSLISTE und den Feld-Objekten jeweils eine Beziehung besteht. Abbildung 14 zeigt das zugehörige Klassendiagramm.

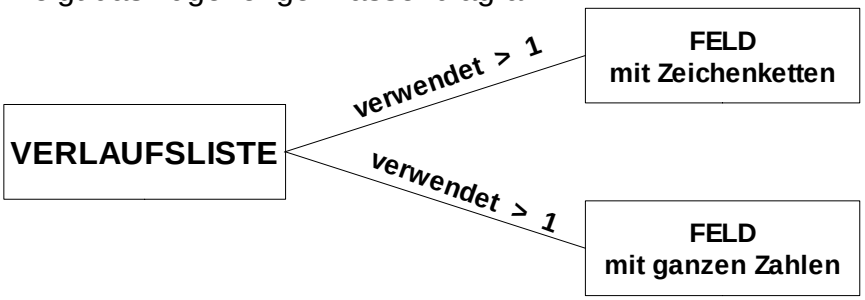


Abbildung 14: Veranschaulichung der Beziehungen im Klassendiagramm der

Im erweiterten Klassendiagramm werden die Felder entsprechend der Kurzschreibweise im Hinweis eingetragen. Die Attribute namen und punkte setzen die Beziehung aus Abbildung 14 um, es sind Referenzattribute, sie verweisen auf die Felder.

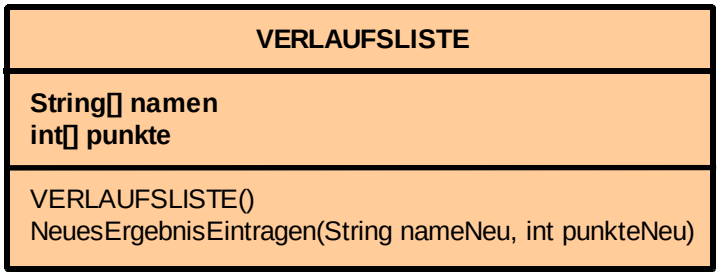


Abbildung 15: Felder am Beispiel der Klasse VERLAUFSLISTE als Referenzattribute im erweiterten Klassendiagramm – erkennbar an dem Paar eckiger Klammern

6.3 Felder in Java umsetzen

Die **Deklaration** eines Datenfeldes sieht in Java folgendermaßen aus:

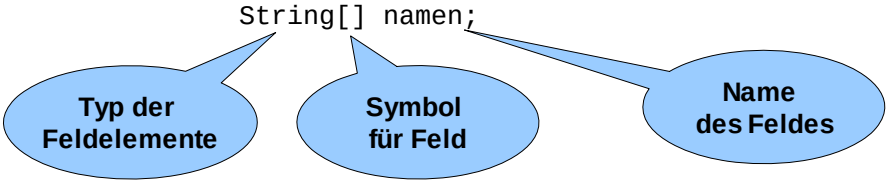


Abbildung 16: Deklaration eines Feldes am Beispiel der Namensliste

Aufgabe 6.10
Überlege dir, was die drei Bestandteile einer Felddeklaration im Modell des Aktenschranks bedeuten. Welche wichtige Information fehlt noch, um den Schrank letztendlich „bauen“ zu können?

Die eckigen Klammern geben an, dass namen nicht der Bezeichner eines einfachen Attributs vom Datentyp String ist, sondern dass es sich um ein Feld handelt, in dem mehrere Attributwerte gespeichert werden können. Aber wie viele genau? Dies wird bei der **Erzeugung** des Datenfeldes festgelegt. Falls Platz zum Speichern von zehn Zeichenketten vorhanden sein soll, lautet die Anweisung dazu:

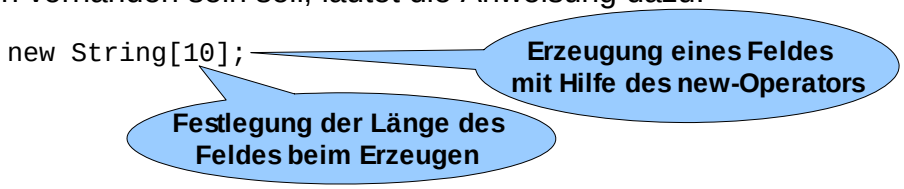


Abbildung 17: Erzeugung eines Feldes mit 10 Feldelementen vom Datentyp String (für die Namensliste)

Nun muss das erzeugte Feld noch dem in Abbildung 16 deklarierten Attribut zugewiesen werden. In Java wird die Erzeugung des Datenfeldes und die **Zuweisung zum Feld-Attribut** in einer Zeile durchgeführt. Für unser Beispiel lautet der Quelltext folgendermaßen:

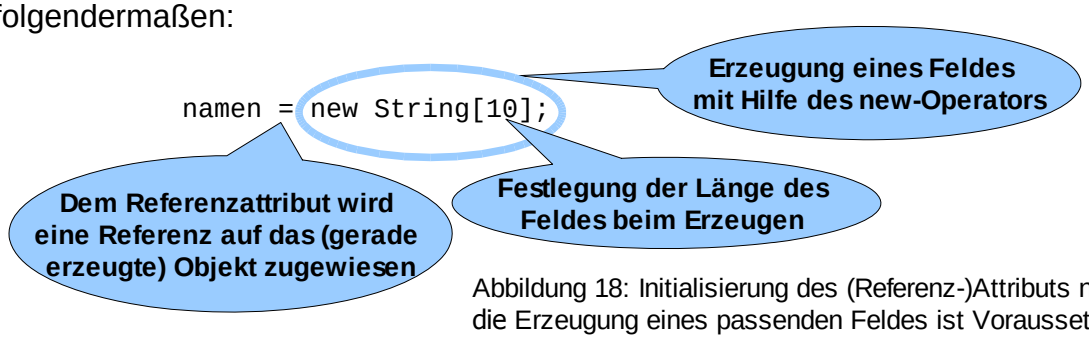


Abbildung 18: Initialisierung des (Referenz-)Attributs namen – die Erzeugung eines passenden Feldes ist Voraussetzung dafür

Der **Zugriff auf einzelne Feldelemente** kann nun einfach über die Angabe des Index in eckigen Klammern erfolgen. Beispielsweise bezeichnet name[3] das Feldelement mit dem Index 3 des Feldes namen (Abbildung19). Im Schrankmodell ist dieses Feldelement die mit 3 beschriftete Schublade des Aktenschranks mit dem Bezeichner namen.

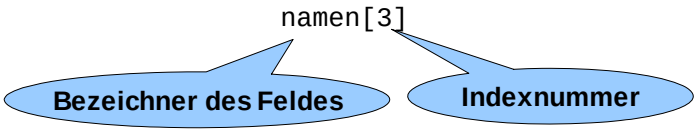


Abbildung 19: Feldelement mit dem Index 3 des Feldes namen

Mit dem Konzept des Datenfeldes reduziert sich die Deklaration innerhalb der Klasse VERLAUFSLISTE deutlich. Für die Namen verringert sich die Anzahl der Quelltextzeilen von zehn (Abbildung 10) auf eine (Abbildung 21).

Hinweis:
Das Vorgehen bei bzw. der Quelltext für die Initialisierung in Abbildung 18 ist identisch zu dem bei Referenzattributen (Abbildung 20).

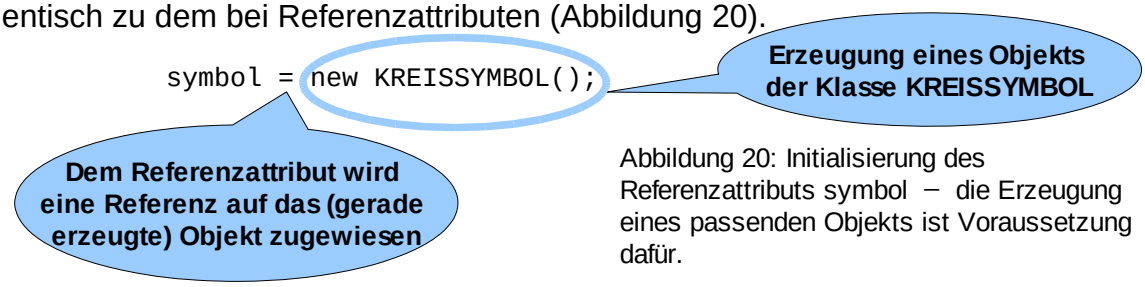


Abbildung 20: Initialisierung des Referenzattributs symbol – die Erzeugung eines passenden Objekts ist Voraussetzung dafür.

class VERLAUFSLISTE

```
class VERLAUFSLISTE
{
   String[] namen;
   // hier fehlt noch die Deklaration des Feldes
   // zum Speichern der Punkte

   VERLAUFSLISTE()
   {
       namen = new String[10];
       namen[0]="---";
       namen[1]="---";
       namen[2]="---";
       namen[3]="---";
       namen[4]="---";
       namen[5]="---";
       namen[6]="---";
       namen[7]="---";
       namen[8]="---";
       namen[9]="---";
       // hier fehlt noch das Erzeugen und
       // Initialisieren des Feldes punkte

   }
   // Methoden
   /*
   void NeuesErgebnisEintragen(String nameNeu,
                               int punkteNeu)
    {
        name3 = name2;
        name2 = name1;
        name1 = nameNeu;

        punkte3 = punkte2;
        punkte2 = punkte1;
        punkte1 = punkteNeu;
    }
    */

}
```

Deklaration

Initialisierung

Beginn eines mehrzeiligen Kommentars

auskommentierte Methode, wird vom Compiler ignoriert

Ende eines mehrzeiligen Kommentars

Abbildung 21: Im Vergleich zu Abbildung 10 ein schon deutlich reduzierter Quelltext der Klasse VERLAUFSLISTE

Aufgabe 6.11

a) Kommentiere als erstes die Methode *NeuesErgebnisEintragen* aus. Sie wird erst in Kapitel 6.3 an die Felder angepasst. Würde sie durch den Compiler überprüft werden, würde er einen Fehler anzeigen. Ein mehrzeiliger Kommentar beginnt mit den Zeichen /* und endet mit */ (siehe Abbildung 20).

b) Setze die Klasse VERLAUFSLISTE entsprechend dem Text mit zwei Feldern der Länge 10 um. (Das erste Feld für die Namen, das zweite für die Punkte.)

c) Erzeuge ein Objekt der in b) geschriebenen Klasse und betrachte mit dem Objektinspektor sowohl das Verlaufslisten-Objekt, als auch die beiden Feldobjekte.
Erkläre knapp, was ein Pfeil im Objektinspektor (z.B. Abbildung 22) bedeutet und woher du ihn schon kennst.

Abbildung 22: Objektinspektor eines Objekts der Klasse VERLAUFSLISTE

6.4 Wiederholung mit fester Anzahl

Von Robot Karol kennst Du bereits das Konzept der „Wiederholung mit fester Anzahl“. Sie dient dazu eine Folge von Anweisungen beliebig oft zu wiederholen.

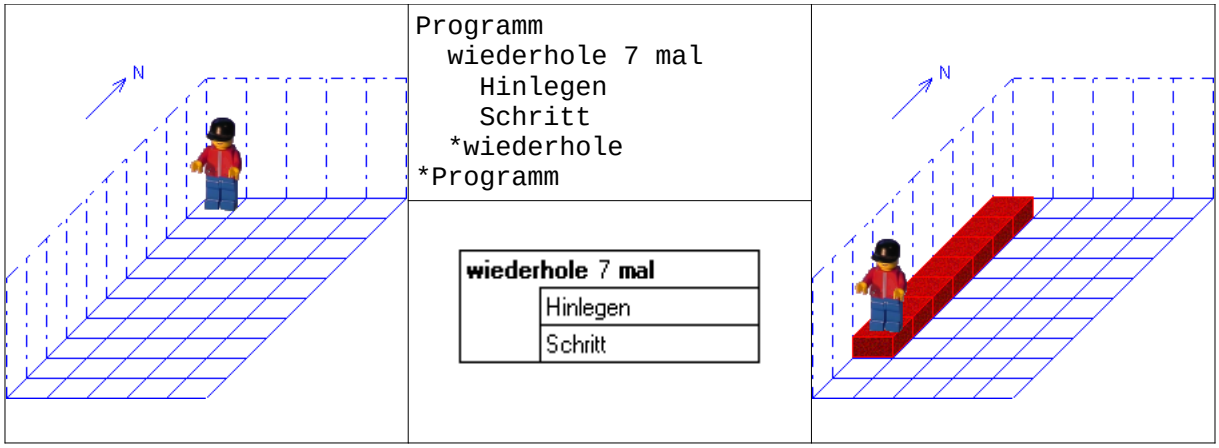


Abbildung 23
Wiederholung mit fester Anzahl bei Karol: Die Welt vorher, nachher, Progarmmtext, Struktogramm

Bei der Initialisierung der Feldelemente des Feldes name werden sehr ähnliche Anweisungen 10 mal wiederholt (Abbildung 21). Hier hilft die „Wiederholung mit fester Anzahl“ den Quelltext zu verkürzen.

Aufgabe 6.12
Die Initialisierung der Elemente des Feldes name der Klasse VERLAUFSLISTE soll mit Hilfe einer Wiederholungsanweisung formuliert werden und nicht wie in Abbildung 20 durch eine Auflistung!
Warum ist es nicht ausreichend, wie bei Robot Karol eine Struktur der folgenden Art zu verwenden?

```
wiederhole 10 mal
     // Sequenz von Anweisungen
* wiederhole
```

Die zu wiederholende Sequenz ist nicht bei jedem Wiederholungsdurchgang identisch, sondern geringfügig verändert.

Aufgabe 6.13
Was ändert sich an der zu wiederholenden Anweisung und was bleibt gleich?
Wie könnte man in einfacher deutscher Sprache die Wiederholung formulieren?

Bei der Initialisierung der Elemente des Feldes name wird die Zuweisung einer leeren Zeichenkette 10 mal wiederholt. Jedoch muss von Zuweisung zu Zuweisung der Index des Feldelements beginnend bei 0 um eins erhöht werden (Abbildung 21). Somit ist folgende Formulierung einer Anweisung mit Wiederholung möglich:

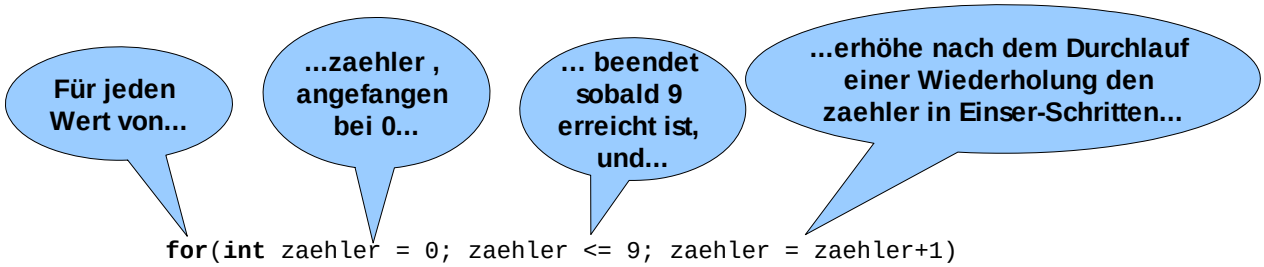
```
Nimm eine Variable zaehler vom Typ integer und setze sie auf 0
Wiederhole 10 mal
          Setze den Wert des Feldelements namen[zaehler] auf die
          Zeichenkette "---".
          Erhöhe den Zähler um eins.
Ende Wiederhole
```

Der zu wiederholende Teil lässt sich in Java deutlich knapper formulieren:

```
namen[zaehler] = "---";
zaehler = zaehler+1;
```

Hinweis:
Die Zuweisung `zaehler = zaehler+1;` kann verwirrend sein, wenn man das Symbol „=“ als das mathematische gleich und nicht die informatische Zuweisung betrachtet. Denke an das Schachtelmodell! Die Zuweisung `zaehler = zaehler+1;` bedeutet: „Der Variablen zaehler wird als neuer Wert ihr alter Wert um eins erhöht zugewiesen.“

Und wie sieht das ganze jetzt in Java aus?
Da die Wiederholungsanweisung mit fester Anzahl in Java eine Zählvariable verwendet, kann diese einfach in der zu wiederholenden Anweisung verwendet werden. Für unser Beispiel lautet die Wiederholungsanweisung:




```
for(int zaehler = 0; zaehler <= 9; zaehler = zaehler+1)
{
      namen[zaehler] = "---";
}
```

Allgemein wird die Wiederholung mit fester Anzahl in Java wie folgt notiert:

Java	Struktogramm

`for(int zaehler = start;`
`  zaehler <= ende;`
`  zaehler = zaehler+schrittweite)`
`{`
`  // zu wiederholende Anweisungen`
`}`
	Zähle zaehler von start bis ende und erhöhe ihn bei jedem Durchlauf um schrittweite
	Anweisungen

Abbildung 24: Wiederholung mit fester Anzahl in Java und als Struktogramm

Vielleicht dauert es ein klein wenig, sich an die Schreibweise der Wiederholungsanweisung zu gewöhnen, aber wegen der großen Arbeitsersparnis weiß man bald sie zu schätzen.

Aufgabe 6.14

a) Optimiere die Klasse VERLAUFSLISTE entsprechend den Erklärungen so, dass zur Initialisierung der Felder punkte und namen eine „Wiederholung mit fester Anzahl“ genutzt wird.

b) Erzeuge ein Objekt der Klasse VERLAUFSLISTE und teste mit dem Objektinspektor, ob die Attributwerte beim Erzeugen richtig gesetzt wurden.

c) Formuliere auch die Methode *NeuesErgebnisEintragen* mit Hilfe der Wiederholungsanweisung. Die in Aufgabe 6.11 eingefügten Kommentarzeichen musst du dazu löschen. (Tipp: Du kannst auch die Zählvariable in jedem Schritt um eins erniedrigen)

d) Erzeuge ein Objekt der Klasse VERLAUFSLISTE und teste die Methode *NeuesErgebnisEintragen,* indem du sie mehrfach mit unterschiedlichen Eingabewerten aufrufst. Beobachte dabei **parallel** in den Objektinspektoren, ob sich die Attributwerte der beiden Felder richtig verändern. Dies bedeutet: Du musst auf deinem Bildschirm drei Fenster so angeordnet haben, dass du sie gleichzeitig sehen kannst:
- das Projektfenster mit der Objektkarte, um die Methode aufrufen zu können (Abbildung 25) und
- die Objektinspektoren der Felder namen und punkte (Abbildung 26)

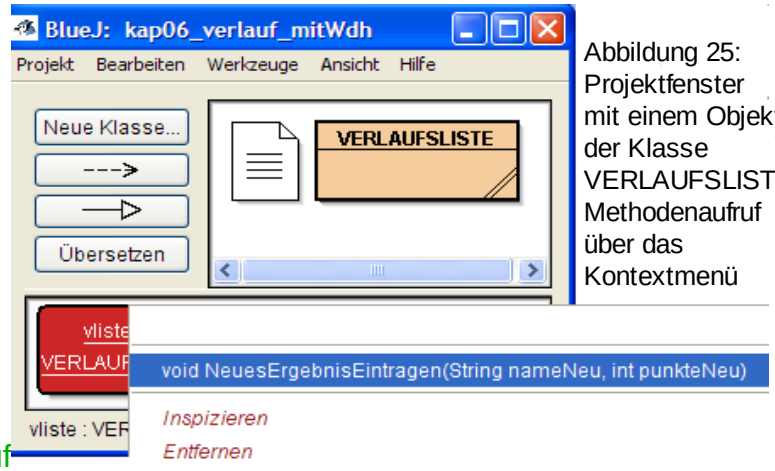


Abbildung 25: Projektfenster mit einem Objekt der Klasse VERLAUFSLISTE; Methodenaufruf über das Kontextmenü

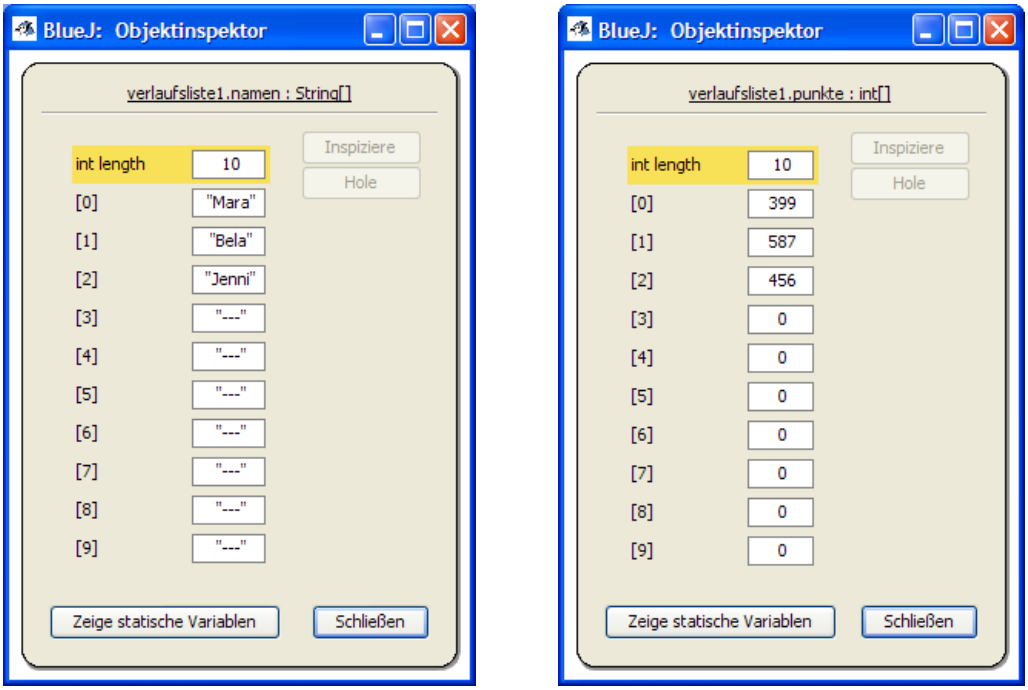


Abbildung 26: Objektinspektoren der Felder name und punkte

6.5 Zusammenfassung

Aufgabe 6.15
Fasse die wesentlichen Inhalte dieses Kapitels in deinem Heft zusammen. Folgende wichtigen Begriffe sollten dabei enthalten sein:
(Daten-)Feld, Feldelement, Index, Deklaration, Initialisierung, Wiederholung mit fester Anzahl, for-Anweisung, Zählvariable

Aufgabe 6.16
Optimiere deine Klasse BESTENLISTE aus Aufgabe 6.9 mit Hilfe von Feldern der Länge 10.

Hinweis:
Die hier im Kapitel vorgestellte Lösung, dass die Punkte unabhängig von den Namen abgespeichert werden und der Zusammenhang über den Index hergestellt wird, ist nicht optimal. Sie wird an späterer Stelle verbessert.

ANHANG
zusätzliche Informationen

Vorgehens-weise

Informatik II: Ein erster Rückblick S. 39

zusätzliche Aufgaben

Das Konzept der Wiederholung mit fester Anzahl lässt sich über folgende Karol-Aufgaben üben:
Informatik II: S. 101/2, 3, 4

Das Konzept des Datenfeldes lässt sich über folgende Karol Aufgabe üben:

Roboter – Eine Ziegelreihe nach Maß

Karol arbeitet in dieser Aufgabe in einer Welt der Länge 5, der Breite 8 und der Höhe 10.

a) Deklariere in der Klasse ARBEIT ein Feld der Länge 8 vom Typ boolean mit dem Namen ZiegelInZeile2. Erzeuge und initialisiere dieses Feldes wie gewohnt im Konstruktor, wobei du die Werte false und true beliebig verteilen darfst (Abbildung 27 zeigt eine mögliche Startbelegung).

Hinweis: Die Initialisierung hier lässt sich hier leider nicht durch eine Wiederholung mit fester Anzahl abkürzen.

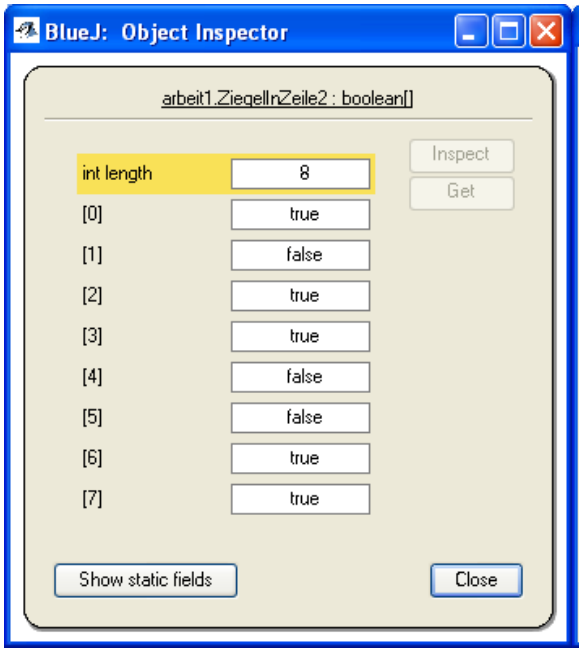


Abbildung 27: Attributwerte der Elemente des Feldes ZiegelInZeile2

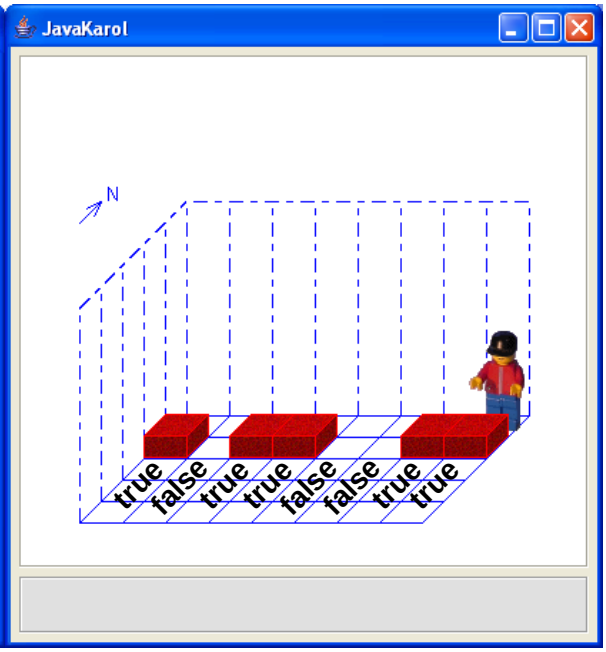


Abbildung 28: Karol hat Ziegel entsprechend der Werte des Feldelemente aus Abbildung 27 gelegt.

b) Schreibe eine Methode *Ausfuehren*, die entsprechend den Werten der Elemente des Feldes ZiegelInZeile2 Karol Ziegel hinlegen lässt (true bedeutet ein Ziegel wird gelegt, false es wird kein Ziegel gelegt) (Abbildung 28).
Hinweise: Verwende eine Wiederholung mit fester Anzahl, achte jedoch darauf, dass Karol nicht an die Wand stösst.

c) Ändere deine Startwerte von den Elementen des Feldes ZiegelInZeile2 (im Konstruktor) und teste, ob deine Methode *Ausfuehren* immer noch richtig arbeitet.

d) Schreibe im gleichen BlueJProjekt eine neue Klasse ARBEIT2. Deklariere dort wieder ein Feld der Länge 8, jedoch diesmal vom Typ int mit dem Namen ZiegelStapelInZeile2. Erzeuge und initialisiere dieses Feldes. Du darfst beliebige Werte zwischen 0 und 10 (Höhe der Welt) wählen.

e) Schreibe eine Methode *Ausfuehren* in der Klasse ARBEIT2, die Karol Ziegelstapel entsprechend den Werten der Elemente des Feldes ZiegelStapelInZeile2 hinlegen lässt.
Hinweis: Eine Schachtelung der Wiederholungsanweisungen führt zum Ziel.