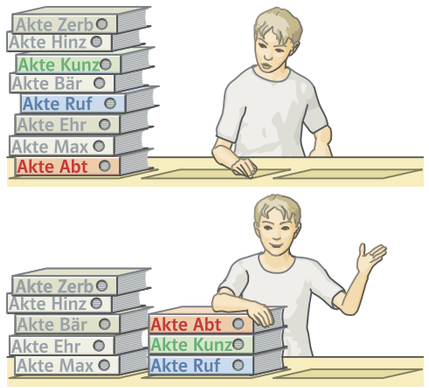
5 Dynamische lineare Datenstrukturen



Peter arbeitet als Auszubildender in einer Anwaltskanzlei und bereitet für seine Chefin die als nächstes zu bearbeitenden Akten vor. Aus einem hohen Stapel sollen die Akten dreier Kunden herausgesucht und alphabetisch in einem neuen Stapel geordnet werden.
Wegen des Gewichts kann immer nur eine Akte verschoben werden. An seinem kleinen Schreibtisch hat Peter insgesamt nur Ablageflächen für drei Aktenstapel.
Wie kann Peter mit möglichst wenig Zügen durch geeignetes Umschichten die Aufgabe lösen?

Bereits in Lerneinheit 1 (Seite 10 ff.) wurden Warteschlangen besprochen, dort jedoch mittels Feldern implementiert.

Einfach verkettete Listen können dazu benutzt werden, Objekte in linearer Anordnung dynamisch zu verwalten. Man bezeichnet sie daher auch als dynamische lineare Datenstrukturen. Je nach Reihenfolge der Entnahme unterscheidet man zwischen **Warteschlange** (First In First Out, kurz FIFO) und **Stapel** (Last In First Out, kurz LIFO).

Die Datenstrukturen Warteschlange und Stapel

Die Herstellung und Verpackung von Schokolade geschieht oft in großen Produktionsanlagen. Am Ende der Produktionslinie kommen dann moderne Verpackungssysteme zum Einsatz, welche zunächst z. B. die unterschiedlichen Sorten der eingepackten Schokoladentafeln einem Förderband zuführen. Dort warten sie auf einen Roboter, der die Tafeln für den nächsten Verpackungsvorgang entnimmt und sie auf einen Stapel legt, von dem beispielsweise einzelne Tafeln zur abschließenden Kontrolle von einer Person entnommen werden (Fig. 1).

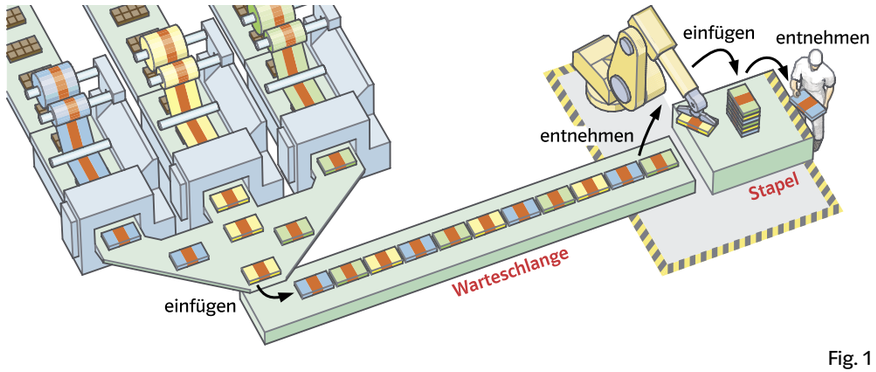


Fig. 1

Die Reihenfolge, in der die Tafeln eingefügt bzw. entnommen werden, entspricht im ersten Fall der einer Warteschlange (vergleichen Sie Seite 10 ff.): Das zuerst aufgenommene Element verlässt als erstes die Liste (FIFO-Prinzip).
Im zweiten Fall wird die zuletzt eingefügte Tafel auch als erste wieder entnommen (LIFO-Prinzip). Man spricht von einem Stapel.

queue *(engl.): Schlange, Reihe*

Der Stapel (engl. stack) wurde bis 1959 von F. L. Bauer an der TU München unter der Bezeichnung Keller entwickelt.

Universelle Einsetzbarkeit von verketteten Listen

Zur Realisierung von Warteschlangen bzw. Stapel verwenden wir die Implementierung verketteter Listen aus Lerneinheit 4. Zur Implementierung der speziellen Methoden von Warteschlange bzw. Stapel werden dann nur noch passende Methoden des ersten Listenelementes aufgerufen.

Auch hier halten wir uns wieder an das Entwurfsmuster Kompositum.

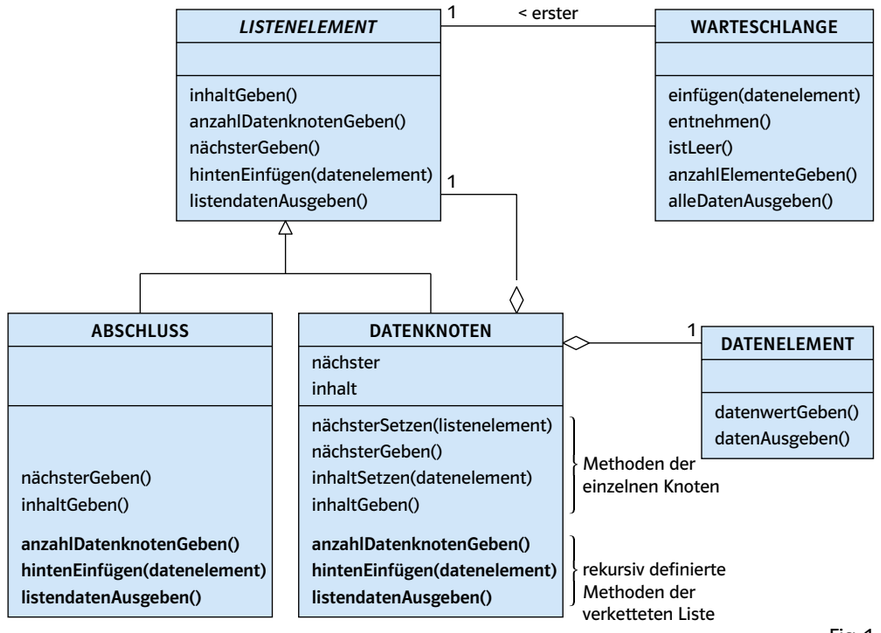


Fig. 1

Für die Warteschlange werden die Methoden *einfügen* bzw. *entnehmen* realisiert, indem in der verketteten Liste hinten eingefügt bzw. vorne entnommen wird.
Das Einfügen erfolgt durch den Aufruf der rekursiven Methode *hintenEinfügen* des ersten Listenelementes, das Entnehmen durch Zuweisung einer Referenz des aktuell zweiten Listenelementes *erster.nächster* an das Attribut *erster* der Warteschlange (Fig. 2).

Rekursiv definierte Methoden der Klasse LISTENELEMENT *und die entsprechende Implementierung in Abschluss und Datenknoten wurden in Lerneinheit 4 konzeptionell behandelt.*

Aus Sicht der Warteschlange spielt es nun keine Rolle mehr, wo vorne bzw. hinten ist.

Java

```
class Warteschlange {
  private Listenelement erster;

  public Warteschlange() {
    erster = new Abschluss();}
  public void einfuegen(Datenelement knoteninhalt) {
    erster = erster.hintenEinfuegen(knoteninhalt);}
  public Datenelement entnehmen() {
    Datenelement alterKnoteninhalt = erster.inhaltGeben();
    erster = erster.naechsterGeben();
    return alterKnoteninhalt;}
  public int anzahlElementeGeben() {
    return erster.anzahlDatenknotenGeben();}
  public void alleDatenAusgeben() {
    erster.listendatenAusgeben();}
  public boolean istLeer() {
    return (anzahlElementeGeben() == 0);}
}
```

Fig. 2