Lineare Datenstrukturen
Hashtabellen

In einer *Hashtabelle* werden *Inhaltobjekte* einem *Schlüsselobjekt* zugeordnet. Die Inhaltsobjekte können dann über das Schlüsselobjekt wieder abgerufen werden.

Beispielsweise sind bei IKEA alle Möbelstücke mit einem Namen und einer (eindeutigen) Artikelnummer versehen. Im Warenlager könnten die Artikel (Inhaltsobjekte) mit ihrer Artikelnummer als Schlüssel gespeichert werden. Wird in der Klasse die Nummer gescannt, kann so schnell auf die Daten des Artikels zugegriffen werden.

Die *generische* Hashtabelle hat die Operationen `put` (Objekt einfügen), `get` (Objekt abrufen) und `remove` (Objekt löschen).

Hashtable<KeyType, ContentType>
-content: ContentType[]
+Hashtable(pSize: int) +put(pKey: KeyType): boolean +get(pKey: KeyType): ContentType +delete(pKey: KeyType) +hasKey(pKey: KeyType): boolean

Aufgabe 1: Das Verhalten einer Hashtabelle kann mit einer einfach verketteten Liste erreicht werden. Formuliere eine Idee, wie die Operationen `put`, `get` und `delete` umgesetzt werden könnten. Welche Laufzeiten haben die Operationen in diesem Fall?

Eine effizientere Methode, die Hashtabelle zu implementieren, basiert auf Arrays:

1. Erstelle ein Array mit einer festen Größe als Speicher.
2. Bestimme einen Index für ein neues Inhaltsobjekt, indem der Hashwert des Schlüsselobjekts modulo der Größe des Arrays gerechnet wird.
3. Speichere das neue Inhaltsobjekt im Array am berechneten Index.

Das Entscheidende hier ist der zweite Schritt: Den Index im Array bestimmen. Ist das Schlüsselobjekt eine Zahl, dann kann direkt die Modulo-Operation (Bestimmung des Rests bei der Division mit Rest) angewandt werden. Handelt es sich um ein anderes Objekt, dann kann in Java mit der Methode `hashCode` eine **eindeutige Zahl für das Objekt** abgerufen werden.

```
int hash = pSchluessel.hashCode();
```

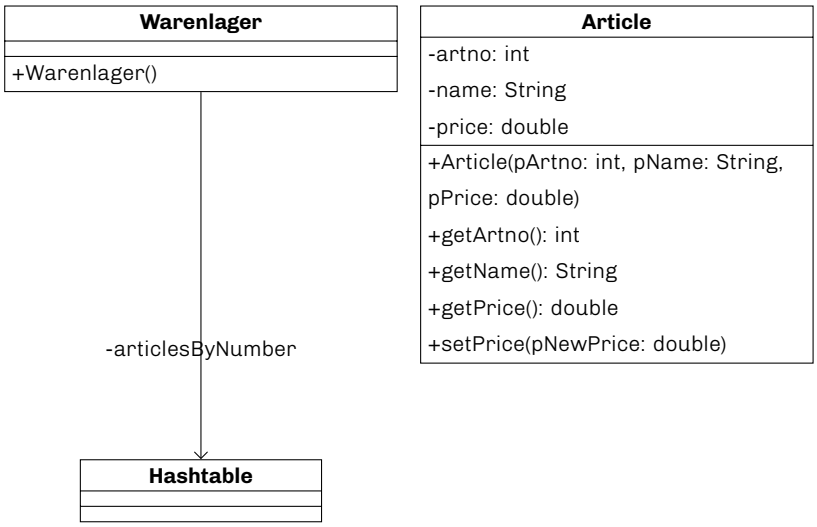
Aufgabe 2: Eine Hashtabelle benutzt ein Array der Größe 31. In welchen Indizes würden folgende Artikel gespeichert, wenn die Artikelnummer als Schlüssel benutzt wird?

Artikelnummer	**Name**	**Index**
158	IVAR	
223	SKADIS	
5	ODGER	

Aufgabe 3: Implementiere eine arraybasierte Hashtabelle nach dem Klassendiagramm oben.

> Das Erstellen eines Arrays mit einem generischen Inhaltstyp ist in Java nicht ohne weiteres möglich. Daher muss zunächst ein unspezifisches Array vom Typ `Object` erstellt werden, das dann in den Typ `ContentType[]` umgewandelt wird.
>
> ```
> content = (ContentType[]) new Object[pGroesse];
> ```

Aufgabe 4: Implementiere das `Warenlager` nach folgendem Klassendiagramm.



Kollisionen

Aufgabe 5: Erstelle eine Hashttabelle der Größe 10 und füge folgende Artikel in die Tabelle ein. Inspiziere nach jedem Artikel den Inhalt des Arrays. Erkläre, was passiert.

Artikelnummer	Name	Index
714	NORDEN	174,49 €
255	NORRAKER	145,24 €
334	INGATORP	252,47 €
217	TUFJORD	535,16 €
115	UTAKER	125,75 €

Aufgabe 6: Informiere dich über Kollisionsvermeidung durch Verkettung | lineares Sondieren | quadratisches Sondieren[1].

Aufgabe 7: Implementiere eine Strategie zur Kollisionsvermeidung in deiner Hashtabelle.

[1] Streiche die Strategien, die du nicht behandelst.