

Motivierendes Beispiel ;-)

- Eine Software soll eine Liste von Schüler:innen verwalten
- Es sollen jederzeit neue Schüler:innen in die Liste aufgenommen und vorhandene entfernt werden können
- Alle Schüler:innen der Liste sollen der Reihe nach ausgegeben werden können

Vorschläge zur Implementierung?

- Zur Speicherung der einzelnen Schüler:innen:
 - Geeignete Klasse definieren
 - Bei Bedarf Objekte erzeugen
- Zur Speicherung der Referenzen auf alle erzeugten Objekte:
 - Array von Referenzen

- Welche Nachteile hat die Verwendung eines Arrays in diesem Fall?
 - Größe nachträglich nicht änderbar
 - Das Einfügen von Schüler:innen “zwischen drin” ist schwierig
 - Das Entfernen von Schüler:innen hinterlässt Lücken

- Wir brauchen eine Datenstruktur, die
 - (beliebig) viele Elemente vom selben Typ speichern kann
 - bei Bedarf wächst oder schrumpft
 - das Einfügen und Löschen an beliebiger Stelle erlaubt

Verkettete Listen

- Erzeuge ein neues Objekt für jedes zu speichernde Element
- “Verkette” die Objekte in der richtigen Reihenfolge miteinander
- Eine solche Datenstruktur heißt daher auch **Verkettete Liste**



Abbildung 1: Verkettete Liste mit vier Objekten

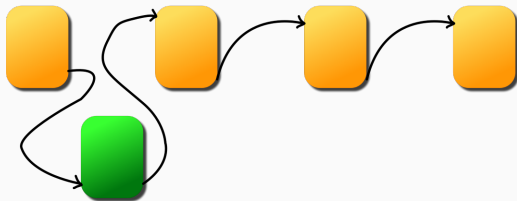


Abbildung 2: Verkettete Liste nach dem Einfügen eines weiteren Objekts

Implementierung in Java

- Hinweis: Zur Vereinfachung vergessen wir die Schulsoftware für's erste und speichern stattdessen nur Werte vom Typ `int` in den Objekten der Liste
- Wie erreichen wir eine Verbindung von einem Objekt zum nächsten?
 - Referenzen
- Implementierungsansatz:

```
class ListElem {  
    private int wert;  
    private ListElem nächstes;  
}
```

`nächstes` ist eine Referenz auf ein weiteres Objekt vom Typ `ListElem`

- Implementiere eine verkettete Liste, bestehend aus:
 - Einer Klasse **ListElem** (s. vorherige Folie) inklusive Settern und Gettern
 - Einer Klasse zur Verwaltung der Liste, die folgendes speichert/implementiert
 - eine Referenz auf das erste Element (Kopf) der Liste
 - die Länge der Liste (Anzahl an Elementen)
 - Methoden zum Anhängen, Einfügen und Löschen von Listenelementen (über eine Angabe der Position)
 - eine Methode zum vollständigen Durchlaufen der Liste (mit Ausgabe aller gespeicherten Werte)

- Erweitere die Implementierung zu einer doppelt verketteten Liste
 - Jedes Listenelement speichert zusätzlich auch eine Referenz auf seinen Vorgänger

- Implementiere Snake
 - Überlege Dir, wieso hier eine verkettete Liste sinnvoll sein könnte
 - Verwende die Klasse `GPanel` oder `Turtle` aus der Bibliothek `aplu.jar` (siehe Moodle)

Und bei der Schulsoftware?

Variante 1: Direkte Verkettung der einzelnen Schüler-Objekte

```
class Schüler {  
    private String name;  
    private Blah blubb;  
    // ...  
  
    private Schüler nächster;  
}
```

- Kommt ohne zusätzliche Objekte aus
- Erfordert Veränderung bei der Schülerklasse
- Unflexibel: Auf Objekte vom Typ **Schüler** beschränkt

Variante 2: Verkettete Hilfsobjekte, die nur auf die Schüler-Objekte verweisen

```
class ListElem {  
    private Schüler s;  
    private ListElem nächster;  
}
```

- Eventuell schwieriger zu verstehen
- Flexibler: Kann so erweitert werden, dass Objekte beliebigen Typs verkettet werden können