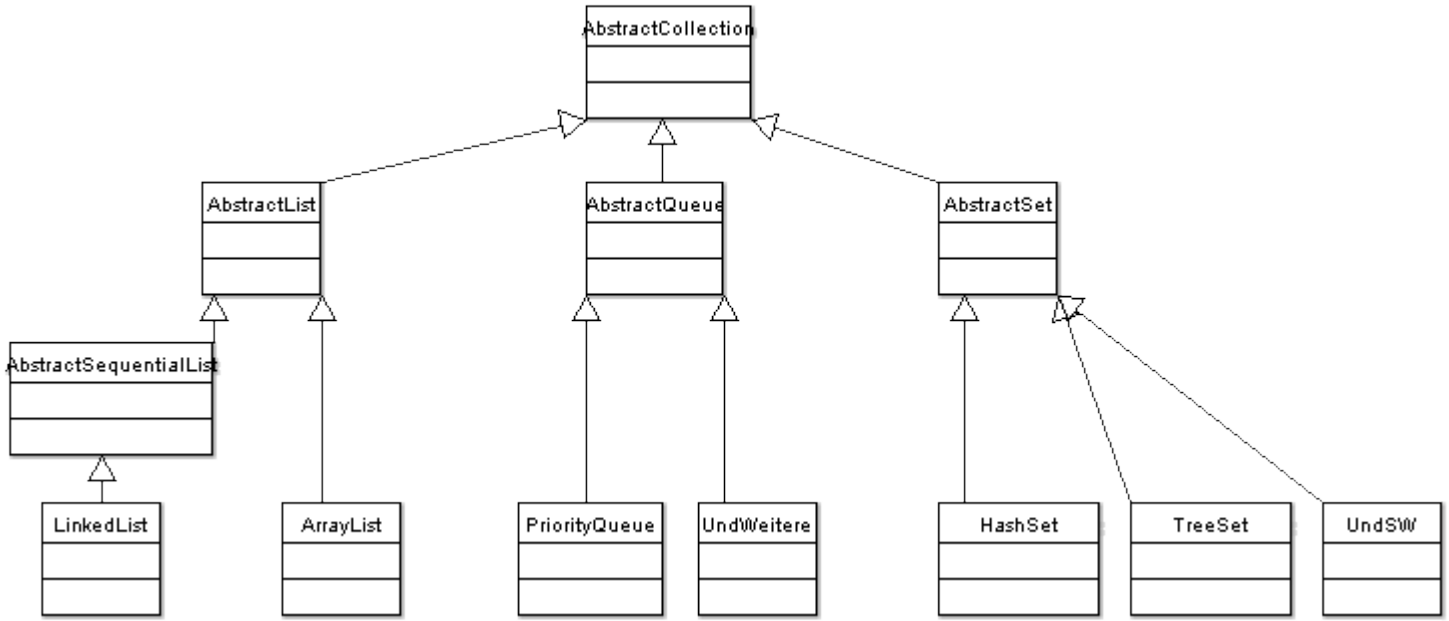
JAVA
Sammlungsklassen

Collections framework

JAVA Sammlungsklassen

- Nur einige ...



JAVA Sammlungsklassen

- Array hat statische Struktur ***(keine Collection)***

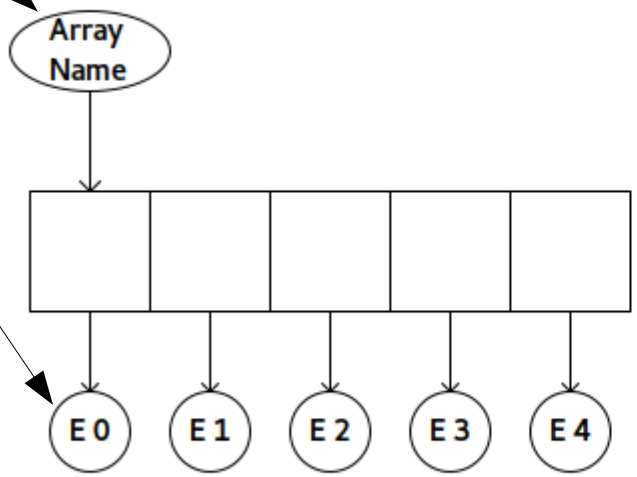
```
Stuhl[] stuehle;
stuehle = new Stuhl[5];
stuehle[0]=new Stuhl();
stuehle[0].zeige();
```

Deklaration

Definition

Definition Inhalt

Zugriff



Referenzen
(Adressen)

JAVA Sammlungsklassen

- Array achtet nicht auf Struktur

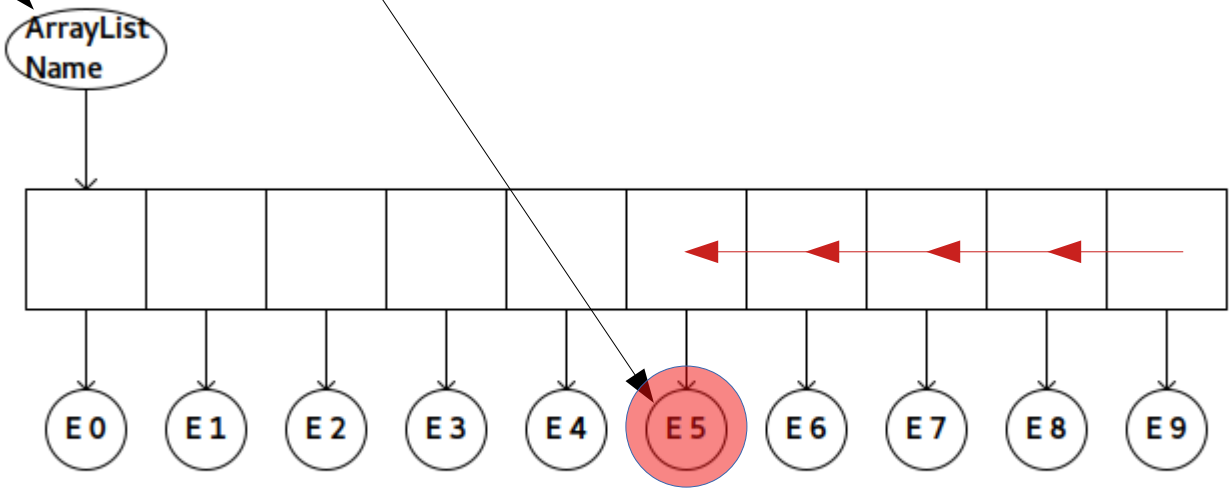
stuehle[5] = null;

löscht nur das Objekt,
belässt die Struktur

Aufrücken gewollt? → „per Hand“

stuehle[5] = stuehle[6];

Schleife für restliche
besetzte Positionen



JAVA Sammlungsklassen

- ArrayList hat dynamische Struktur

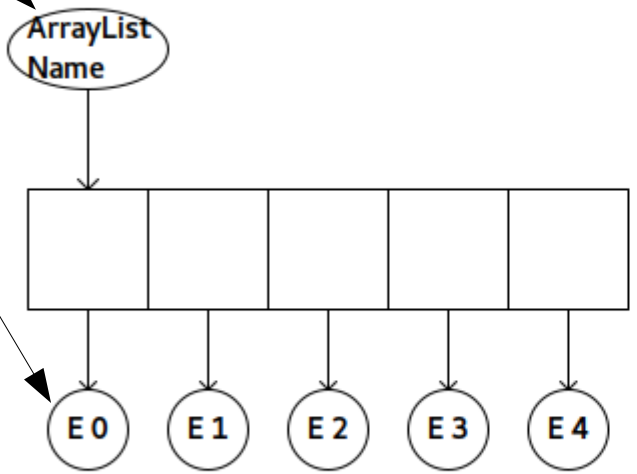
```
ArrayList<Stuhl> stuehle;
stuehle = new ArrayList();
stuehle.add( new Stuhl() );
stuehle.get(0).zeige();
```

Deklaration

Definition

Definition Inhalt

Zugriff



Referenzen
(Adressen)

JAVA Sammlungsklassen

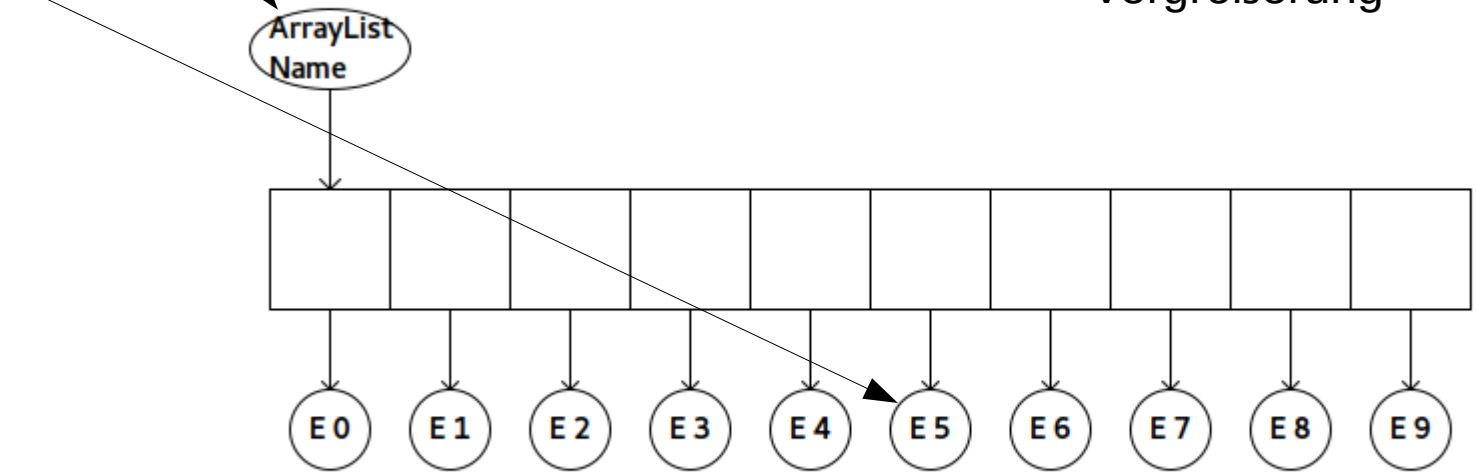
- ArrayList hat dynamische Struktur

stuehle.append(... fuellen

stuehle.size() → 5 also voll

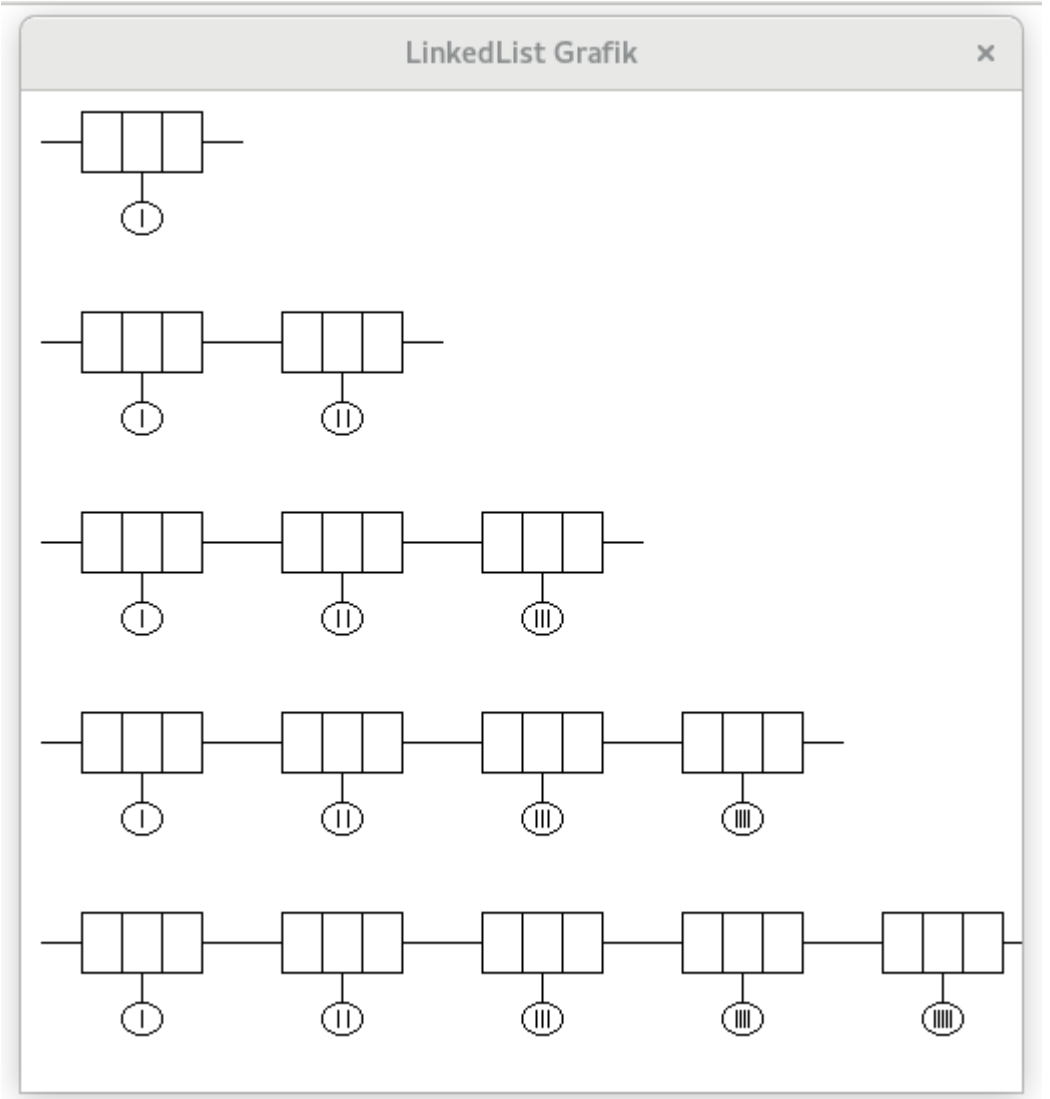
stuehle.add(new Stuhl()); noch ein Stuhl geht aber !

stuehle.get(5).zeige(); Automatische Vergrößerung



JAVA Sammlungsklassen

- LinkedList
- Bei Java
  mit Zeiger auf
  Vorgänger
  und Nachfolger



JAVA Sammlungsklassen

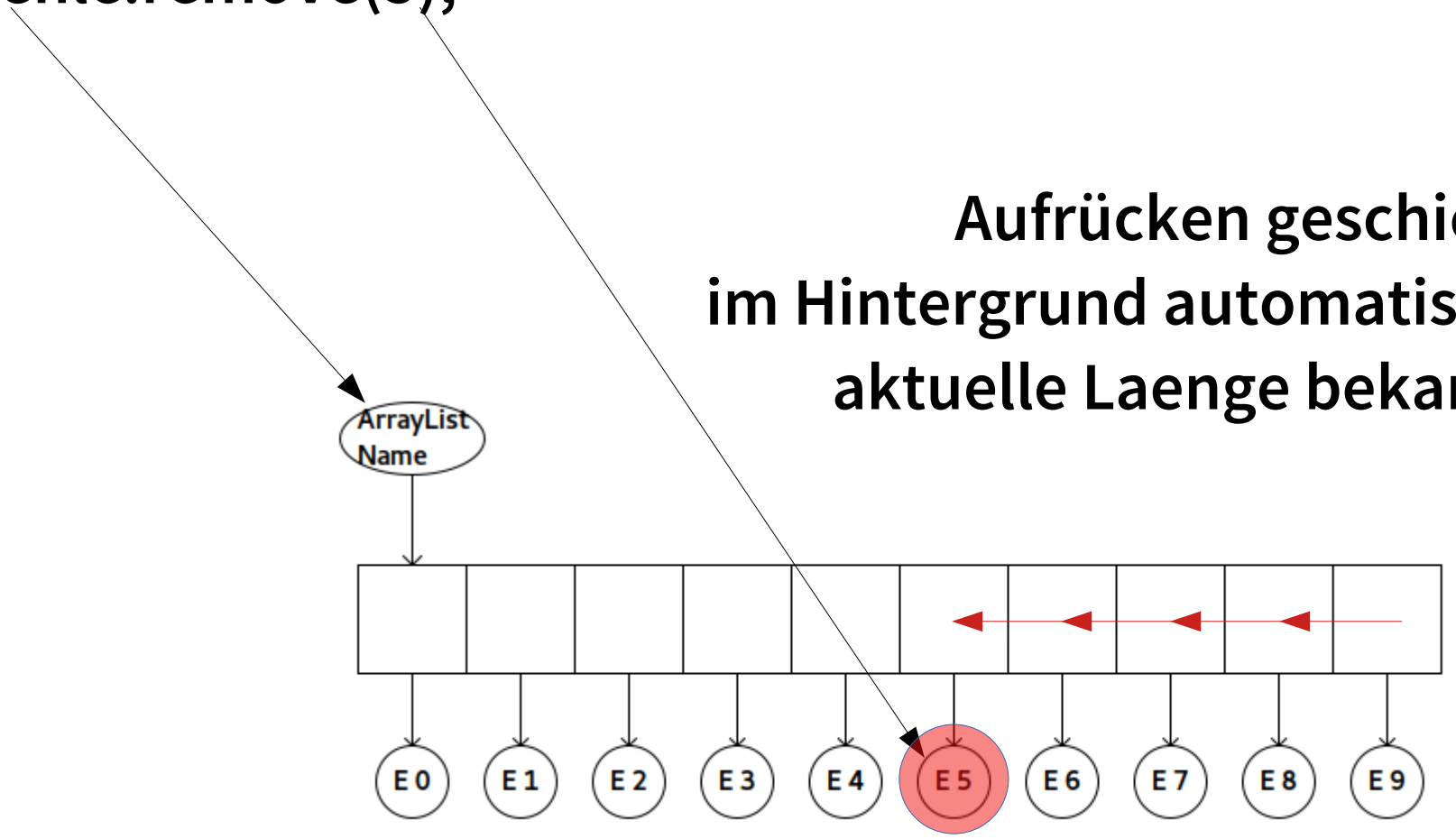
Vorteile / Nachteile

- ArrayList
  - Zugriffe typisch Index-basiert und schnell
  - Einfügen / Löschen im Inneren aufwändig
- LinkedList
  - Zugriffe prinzipiell sequenziell, ggf aufwändig
  - Einfügen / Löschen sehr einfach
- beide
  - gehören zum Collections Framework und stellen dessen typische Methoden bereit
  - implementieren das Interface List

JAVA Sammlungsklassen

- Löschen bei ArrayList

stuehle.remove(5);



Aufrücken geschieht im Hintergrund automatisch, aktuelle Laenge bekannt

JAVA Sammlungsklassen

- Löschen bei LinkedList

stuehle.remove(2);

Umsetzen der Zeiger („Verbiegen“)
geschieht im Hintergrund automatisch