Javakurs für Anfänger

Einheit 08: Mehr zu Arrays

Kyrill Schmid
Lehrstuhl für Mobile und Verteilte Systeme

Heutige Agenda

1. Teil: Wiederholung und Fragen zu Arrays

- 2-Dimensionale Arrays am Beispiel Schachbrett

2. Teil: Arrays anpassen

- Arrays kopieren
- Elemente
  - Einfügen
  - Löschen
  - Finden
- Dynamische Arrays mit der `ArrayList`

Praxis:

- Übungen mit Arrays von unbekannter Größe
  - Nutzer tätigt Eingaben

Lernziele

- Den Umgang mit Arrays weiter einüben
- Arrays anpassen und für verschiedene Situationen nutzen lernen
- Die Klasse `ArrayList` kennenlernen

Übungsaufgaben zu 2-dimensionalen Arrays

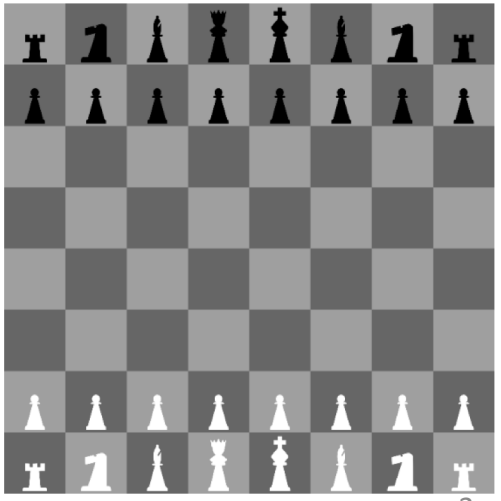
Aufgabe (aus der letzten Kursstunde):

Schreiben Sie ein Programm „Schachbrett“ das folgende Aufgaben erledigt:

- Das Programm soll ein Schachbrett mit Grundaufstellung (also vor dem ersten Spielzug) simulieren und auf der Konsole ausgeben, siehe Abbildung rechts →

```
Console
<terminated> Schachbrett [Java Application]
T S L D K L S T
B B B B B B B B
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
B B B B B B B B
T S L D K L S T
```

- Das Schachbrett soll als 8x8 `char`-Array gespeichert sein.
- Jedes Feld des Schachbretts, auf dem **keine** Spielfigur steht, zeigt eine **0** an
- Jedes Feld des Schachbretts, auf dem **eine** Spielfigur steht, zeigt den **Anfangsbuchstaben** der jeweiligen Spielfigur (ohne Farbe) an:
  - K=König, D=Damen, L=Läufer,
  - S=Springer, T=Turm und B=Bauer
- Verwenden Sie passende Kontrollstrukturen und achten Sie auf eine effiziente Programmierung!



Arrays anpassen

Wiederholung:

- Die Größe eines Arrays kann nach dessen Definition **nicht mehr verändert** werden

Häufig weiß man aber zum Zeitpunkt der Array-Erzeugung nicht, wie groß es sein muss.

- Bsp.: Nutzereingaben werden in Array gespeichert. Aber wie viele Eingaben tätigt ein Nutzer?

Lösungen:

- Man erzeugt ein sehr großes Array und befüllt es dann nur teilweise
  - Speicherplatzverschwendung
  - Reicht die gewählte Größe wirklich aus?
    => Fehlerbehandlung

- Verwendung einer dynamischen Array-Struktur
  - `ArrayList` aus dem Paket `java.util`
  - Kommt am Ende dieser Stunde

Arrays kopieren

Sollte der Platz eines Arrays nicht ausreichen, kann dann auch ein größeres erzeugt werden.

- Die Elemente des alten Arrays müssen kopiert werden!
- Normalerweise doppelte Größe für das neue wählen

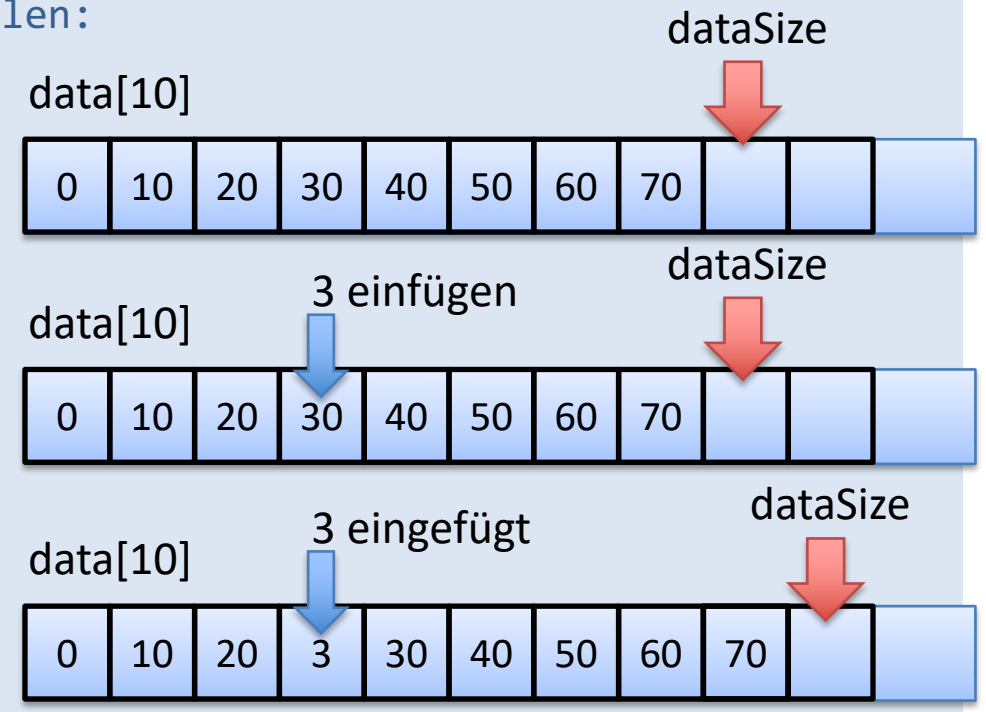
Um ein Array tatsächlich zu kopieren, muss jedes Element in ein neues Array geschrieben werden

```
// Array kopieren
double[] temperaturen_kopie = new double[temperaturen.length*2];

for (int i = 0; i<temperaturen.length;i++){

    temperaturen_kopie[i] = temperaturen[i];

}
```

Arrays anpassen: Element Einfügen

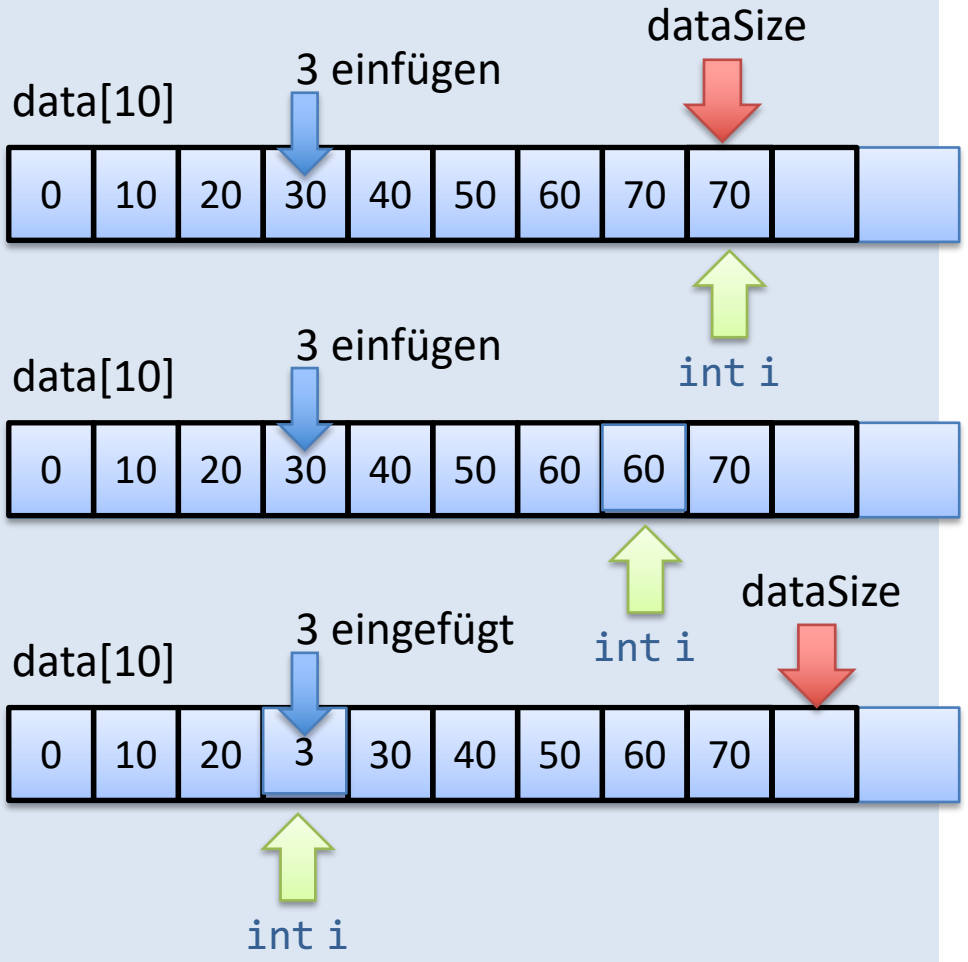
Man will ein Element an einer bestimmten Stelle pos in ein teilweise befülltes Array einfügen und die Ordnung beibehalten.

```
// Beispiel: Einfügen der Zahl 3 an der 4. Stelle eines Arrays data[10]

// Array erzeugen und teilweise befüllen:
int[] data = new int[10];
int dataSize = 8;
for(int i=0;i<dataSize;i++)
   data[i]=i*10;

// Zahl 3 an der 4. Stelle einfügen
//und die Ordnung beibehalten!

int pos=3;
for (int i = dataSize; i > pos; i--)
   data[i] = data[i-1];
data[pos] = 3;
dataSize = dataSize + 1;
```



Element einfügen im Detail


```
// Nochmal im Detail: Element einfügen mit Beibehaltung der Ordnung:

int pos=3;
for (int i = dataSize; i > pos; i--)
   data[i] = data[i-1];
data[pos] = 3;
dataSize = dataSize + 1;
```



Nach letztem Schleifendurchlauf:

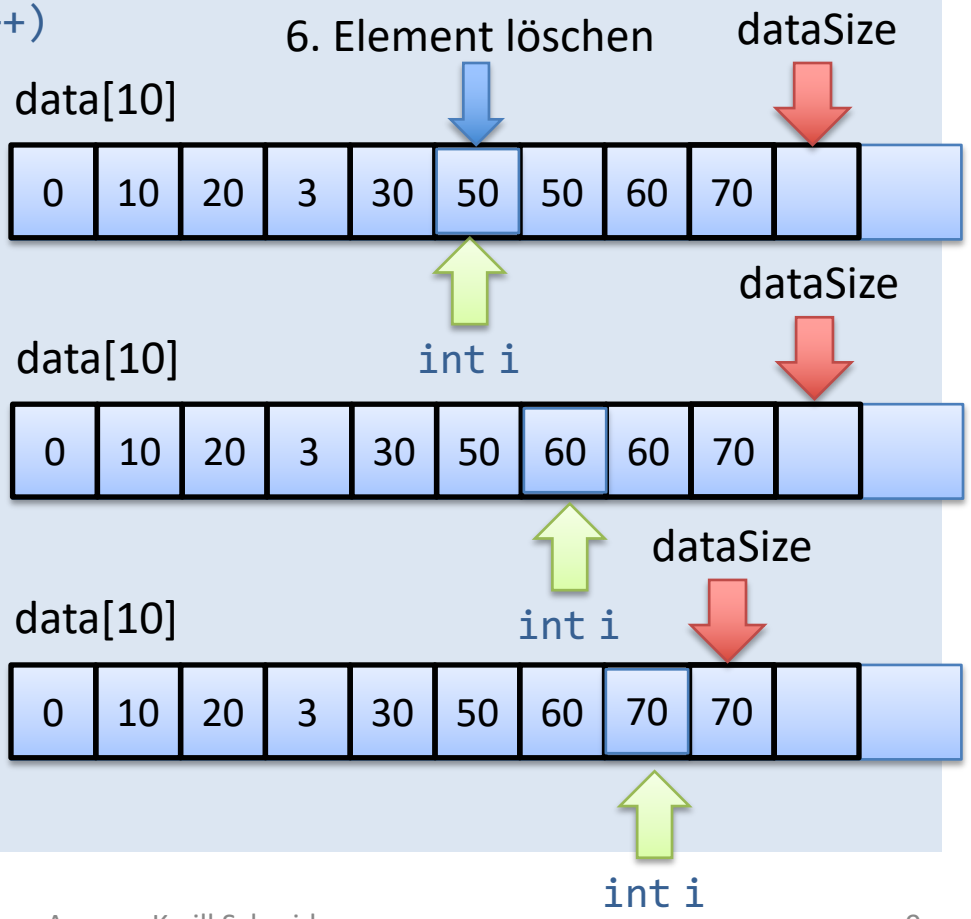
Arrays anpassen: Element Löschen

Man will ein Element an einer bestimmten Stelle pos in einem teilweise befülltem Array löschen:

```
//Falls die Ordnung keine Rolle spielt:

data[pos] = data[dataSize-1]; // Letztes Element ersetzt den Wert bei pos
dataSize = dataSize - 1; // Die Datengröße wird um eins verringert


// Falls die Ordnung beibehalten werden muss:
for (int i = pos; i < dataSize - 1; i++)
   data[i] = data[i+1];
dataSize = dataSize - 1;
```

Element löschen im Detail


```
// Nochmal im Detail: Element Löschen mit Beibehaltung der Ordnung
// Beispiel Löschen des 6. Elements:

int pos= 5;
for (int i = pos; i < dataSize - 1; i++)
   data[i] = data[i+1];
dataSize = dataSize - 1;
```



Arrays anpassen: Element Finden

Man möchte Elemente in einem Array finden, die bestimmte Eigenschaften erfüllen.

- Um dann evtl. diese Elemente zu verändern (überschreiben)
- Oder, um zu wissen, dass solche Elemente existieren (lesen)

```
// Beispiel: Finden aller Autos, die teurer sind als 20.000, um dann einen
1000 Euro Rabatt zu gewähren:

for (Auto mein_auto : meine_autos){
   if(mein_auto.getPreis()>20.000)
      mein_auto.setPreis(mein_auto.getPreis()-1000);
}

// Oder überprüfen, ob es eine Temperatur gibt, die kleiner war als 1,0 Grad.

boolean gefunden = false;
for (i = 0; i < temperaturen.length; i++)
   gefunden = gefunden || (temperaturen[i] < 1.0);

// boolean gefunden bleibt true, sobald (temperaturen[i] < 1.0) einmal true
wurde
```

Programmieraufgabe

Aufgabe 1: Nutzereingaben speichern

Erstellen Sie in Ihrem Eclipse-Projekt „Uebung09“ ein neues Programm „Nutzereingaben“, das folgende Aufgaben erledigt:

- Der Nutzer wird immer wieder aufgefordert eine Zahl einzugeben, solange bis eine 0 eingegeben wird.
- Das Programm speichert die Nutzereingaben (Ohne die zur Terminierung eingegebene 0) in eine passende Datenstruktur.
- Anschließend berechnet das Programm den Durchschnitt der eingegebenen Zahlen und gibt das Ergebnis auf der Konsole aus.
- Beachten Sie dabei bitte folgende Punkte:
  - Sie müssen überprüfen, ob eine eingegebene Zahl noch in Ihrer Datenstruktur gespeichert werden kann, bevor Sie die Zahl speichern!
  - Wenn das nicht mehr der Fall ist, müssen Sie sich überlegen, wie Sie mit der Nutzereingabe umgehen wollen:
    - Entweder stellen Sie mehr Speicher zur Verfügung und speichern die Zahl
    - Oder Sie sagen dem Nutzer, dass seine Zahl nicht mehr gespeichert werden kann und beenden die Aufforderung zur Zahleneingabe und fahren im Programm fort.

Arrays anpassen

Wiederholung:

- Die Größe eines Arrays kann nach dessen Definition nicht mehr verändert werden

Häufig weiß man aber zum Zeitpunkt der Array-Erzeugung nicht, wie groß es sein muss.

- Bsp.: Nutzereingaben werden in Array gespeichert. Aber wie viele Eingaben tätigt ein Nutzer?

Lösungen:

- Man erzeugt ein sehr großes Array und befüllt es dann nur teilweise
  - Speicherplatzverschwendung
  - Reicht die gewählte Größe wirklich aus?
    => Fehlerbehandlung

- **Verwendung einer dynamischen Array-Struktur**
  - `ArrayList` aus dem Paket `java.util`

Dynamische Arraystruktur

Die Array-Liste: `java.util.ArrayList`

- Sehr eng mit Arrays verwandt
  - Elemente sind wie bei Arrays linear angeordnet
  - Können mit Index (0 bis length-1) angesprochen werden
  - Ein Objekt der `ArrayList` wird mit `new` erzeugt
- Aber:
  - Elemente können dynamisch hinzugefügt und/oder entfernt werden
  - Keine Angabe einer vordefinierten Länge nötig
  - Nur komplexe Datentypen (Referenztypen) erlaubt!
    - Um dennoch primitive Datentypen zu nutzen werden Wrapperklassen verwendet
      - `Double`, `Integer`, usw.

Erzeugen von `ArrayLists`

Zunächst muss wieder die Klasse `ArrayList` aus dem Packet `java.util.ArrayList` importiert werden!

Bei der Deklaration einer `ArrayList` muss ein Typ der Elemente mit angegeben werden:

- Dieser steh in spitzen Klammern `<Typ>`
  - Stichwort: „generics“ wird später behandelt
- Allgemeine Syntax zur Erzeugung einer ArrayList:
  - `ArrayList<Datentyp> name = new ArrayList<Datentyp>();`

```
// Beispiele zum Erzeugen von ArrayLists verschiedener Datentypen:

ArrayList<Double> temperaturen = new ArrayList<Double>();

ArrayList<Integer> eingabeZahlen = new ArrayList<Integer>();

ArrayList<Auto> meine_autos = new ArrayList<Auto>();
```

Arbeiten mit `ArrayList`

Auf einem `ArrayList`-Objekt können verschiedenste Methoden angewendet werden. Nähere Infos hierzu in der Java API Spezifikation

- http://docs.oracle.com/javase/8/docs/api/

Die wichtigsten Methoden sind in der Tabelle jeweils mit Beispiel dargestellt:

- Angenommen es existiert eine `ArrayList<Auto> autoListe`
- Und eine `ArrayList<Double> temperaturen;`

Methode (allgemein)	Funktion	Beispiele
`void clear()`	Löscht alle Elemente aus der `ArrayList`	`autoListe.clear();` `temperaturen.clear();`
`E get(int index)`	Liefert das Element vom Typ `E` an der Position `index` zurück	`Auto a = autoListe.get(2);` `double b = temperaturen.get(5);`
`E set(int index, Object o)`	Ersetzt das Element `E` an der Position `index` durch das übergebene Objekt `o`	`autoListe.set(2,fiat);` `temperaturen.set(3,19.8);`
`int size()`	Liefert die Anzahl der Elemente in der Liste zurück	`int anzahlAutos = autoListe.size();`

Arbeiten mit `ArrayList`

Methode (allgemein)	Funktion	Beispiel
`int indexOf(Object e)`	Liefert den Index des übergebenen Elements zurück. Wenn nicht vorhanden, dann wird -1 zurückgeliefert.	`int tag = temperaturen.indexOf(23.3);` `int positionMercedes = autoListe.indexOf(mercedes);`
`E remove(int index)`	Entfernt das Element E an der Position index aus der ArrayList	`autoListe.remove(2);` `temperaturen.remove(5);`
`void add(int index, Object o)`	Fügt der Liste an der Position index das übergebene Objekt als neues Element hinzu	`autoListe.add(3,mercedes);` `temperaturen.add(7,23.5);`
`boolean add(Object o)`	Fügt der Liste das Objekt am Ende hinzu	`autoListe.add(fiat);` `temperaturen.add(22.7);`

Programmieraufgabe

Aufgabe 2: Verwendung der Klasse `ArrayList`:

Schreiben Sie nun Ihr zuvor erstelltes Programm „Nutzereingaben“ so um, so dass Sie statt fester Arrays nun ein Objekt der Klasse `ArrayList` nutzen, welches in der Lage ist, die Eingaben des Nutzers dynamisch zu speichern.

- Nach Eingabe einer 0 soll das Programm die eingegebenen Werte in der `ArrayList` wieder durchgehen und den Mittelwert berechnen, der anschließend auf der Konsole ausgegeben wird.

Bis nächstes Jahr…

Frohe Weihnachten und an guadn Rutsch ins neue Jahr!

Wir sehen uns am 09. Januar wieder!