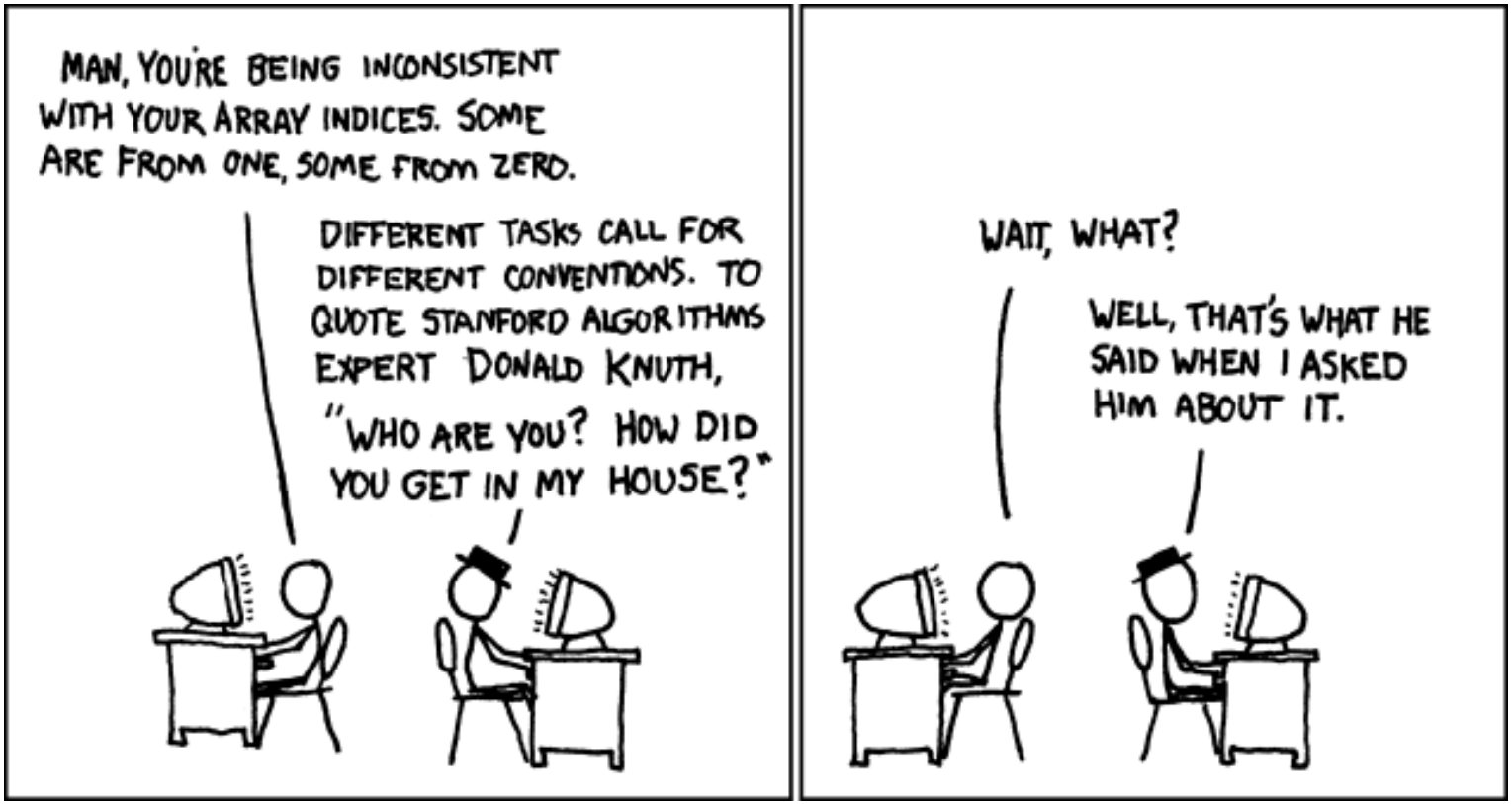
Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Listen

Vorkurs Informatik

Universität Stuttgart
Karlsruher Institut für Technologie
F. Schweiner
MINT-KOLLEG BADEN-WÜRTTEMBERG

Getting started ...



©Randall Munroe, `http://xkcd.com/163/`

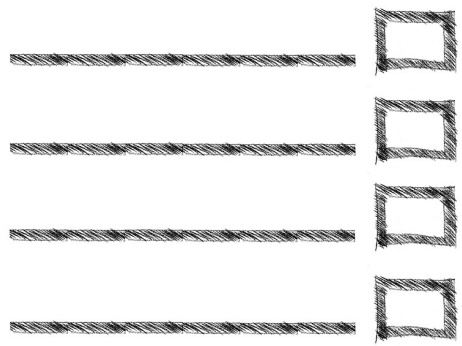
Outline

(nach Folien von L. Vettin, V. Weidler, W. Kessler)

Outline

Listen

- Wir betrachten nun die Klasse `ArrayList<E>`.
- Objekte dieser Klasse sind Listen, wie man Sie beispielsweise von Textdokumenten kennt.



https://pixabay.com/de/photos/rechteck-liste-linien-einkaufsliste-2470300/

Listen

- Listen können versch. Objekte speichern (Texte, Zahlen, ...).
- Hierfür schreibt man den gewünschten Typ in spitze Klammern, z.B. `ArrayList<String>`.
- Ein Objekt `myList` dieser Klasse könnte wie folgt aussehen:

Listen

- Listen können versch. Objekte speichern (Texte, Zahlen, ...).
- Hierfür schreibt man den gewünschten Typ in spitze Klammern, z.B. `ArrayList<String>`.
- Ein Objekt `myList` dieser Klasse könnte wie folgt aussehen:

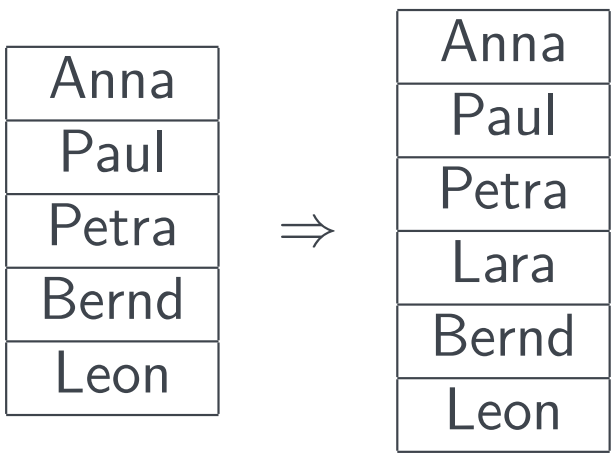
- Jedem Element ist eine ganze Zahl (index) zugeordnet.
- Diese startet bei 0 und besitzt als Maximalwert „Länge - 1“.
- Beispielsweise hat obige Liste die Länge 5.
- Zu Anna gehört der Wert 0, zu Leon der Wert 4.

Listen

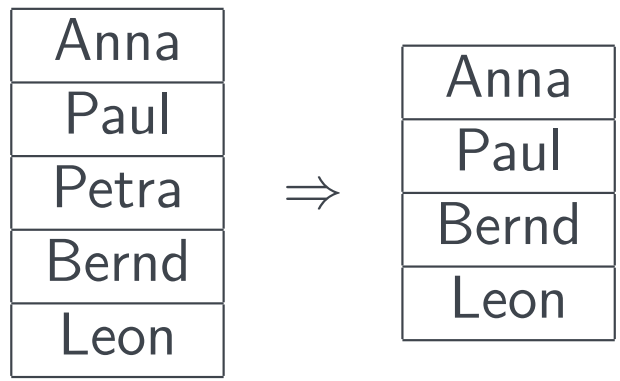
- Welche Operationen könnte eine Liste bieten?
- Welche Argumente benötigen die Operationen?
- Welche Operationen sind Kommandos? Welche Abfragen?

Beispiel: Listen

- Zum Hinzufügen und Entfernen von Elementen gibt es z.B. die Operationen `add` und `remove`.
- Hinzufügen: Wo und was soll eingefügt werden?
- Entfernen: Wo soll etwas gelöscht werden?




```
myList.add(3,"Lara");
```




```
myList.remove(2);
```

- Mit der Operation `size` lässt sich die Größe einer Liste abfragen. Sie liefert einen Wert zurück und ist also eine Abfrage.

Listen

Liste

Eine Liste ist eine Datenstruktur, die mehrere Objekte des gleichen Datentyps zusammenfasst.

Listen

Liste

Eine Liste ist eine Datenstruktur, die mehrere Objekte des gleichen Datentyps zusammenfasst.

- Listen sind Objekte und können nur Objekte enthalten.
- Listen haben eine variable Länge. Es können flexibel zu jedem Zeitpunkt Objekte hinzugefügt oder gelöscht werden.
- Listen werden Element für Element gespeichert. Um auf ein bestimmtes Element zuzugreifen, wird immer die ganze Liste durchlaufen.
- Listen sind im Paket `java.util` implementiert.

Generics

Generischer Typ, parametrisierter Typ

Ein generischer Typ ist ein Typ, der Platzhalter für andere, unbekannte Datentypen enthält. Bei der Verwendung muss für den Platzhalter ein konkreter Datentyp angegeben werden.

- Der unbekannte Typ gibt bei `List` den Typ der Elemente an, die in der Liste gespeichert werden.
- Der konkrete Typ muss bei der Verwendung in spitzen Klammern hinter `List` angegeben werden:
  - `List<String>` ($\rightarrow$ Liste von Strings)
  - `List<Raccoon>` ($\rightarrow$ Liste von Objekten vom Typ `Raccoon`)
- Generics gehören zu den schwierigsten Themen in Java. Wir behandeln hier nur die Verwendung von generischen Listen.

Generics

- Als Typ-Parameter dürfen nur Objekt-Datentypen verwendet werden, d. h. einfache Datentypen (int, double, char, . . .) dürfen nicht verwendet werden.
- Als Ersatz hierfür gibt es die Wrapper-Klassen

Generics

- Als Typ-Parameter dürfen nur Objekt-Datentypen verwendet werden, d. h. einfache Datentypen (int, double, char, . . .) dürfen nicht verwendet werden.
- Als Ersatz hierfür gibt es die Wrapper-Klassen

Wrapper

Wrapper-Klassen sind Klassen, die dazu dienen, eine Anweisungsfolge oder einen einfachen Datentyp in die Gestalt einer Klasse zu bringen.

- Für alle einfachen Datentypen gibt es in Java im Paket java.lang die folgenden Wrapperklassen:

  `Character, Boolean, Byte, Short, Integer, Long, Double, Float`

- Die Wrapper-Klassen stellen Methoden zur Bearbeitung der Datentypen bereit. Beispiele hierfür sind Methoden zum Umwandeln von Zahlen in Strings und von Strings in Zahlen

Listen erstellen

- Liste erstellen: `List<Typ> myList = new ArrayList<Typ>();`
- Elemente können mit `add` hinzugefügt werden.
- Der Datentyp der eingefügten Elemente muss passen.

```
List<String> myList = new ArrayList<String>();
myList.add("Hello");
myList.add("World!");
myList.add("!");
myList.add(fluffy); // Fehler!
```

Listen verwenden

- Der Zugriff auf die einzelnen Elemente erfolgt über einen Index mit der Methode `get`. Indizes gehen von 0 bis Länge-1.
- Ist der Index nicht vorhanden, gibt es einen Fehler:

```
Exception in thread "main"
   java.lang.IndexOutOfBoundsException: 4
```

- Die Länge einer Liste wird mit der Methode `size` abgefragt.

```
System.out.println( myList.get(0) );
System.out.println( "length = " + myList.size() );
System.out.println( myList.get( myList.size() ) ); // Fehler!
```

Listen im Code

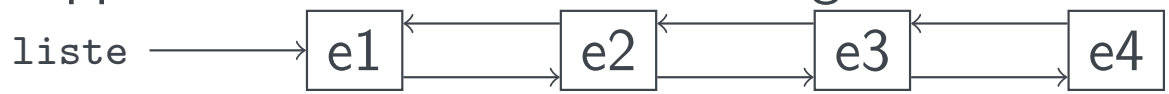
```java
// Liste anlegen
List<Integer> myList = new ArrayList<Integer>();

// Werte der Listen-Elemente setzen
myList.add(6);
myList.add(2);
myList.add(13);

// Wert eines BESTEHENDEN Elements aendern
// hier vom 3. Element auf den Wert 6
myList.set(2,6);

// Laenge, Elemente, ganze Liste ausgeben
System.out.println("length: " + myList.size()); // 3
System.out.println("second element: " + myList.get(1)); // 2
System.out.println(myList);
```

- Mit `Arrays.asList` lässt sich eine Liste in einem Schritt füllen.
- Dann kann die Listengröße aber nicht mehr geändert werden!

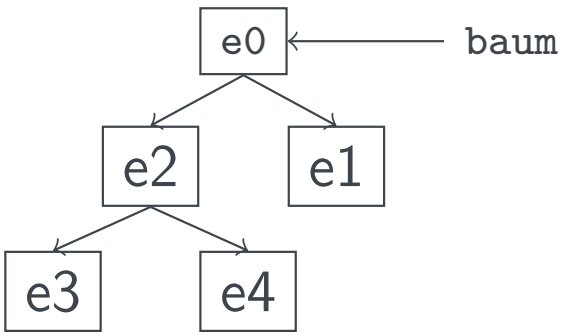
```java
List<Integer> myList = Arrays.asList(6, 2, 13);
```

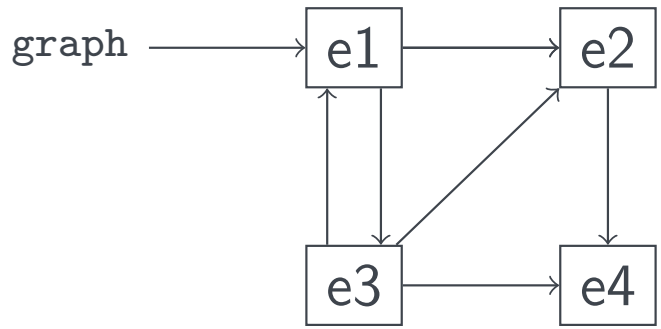
Andere Datenstrukturen

- Mit listenähnlichen Klassen lassen sich auch andere Datenstrukturen implementieren.
- Doppelt-verkettete Liste: Nachfolger `next` und Vorgänger `last`.



- Binärbaum: 2 Kinder `left` und `right`.





- Graph: eine Liste von ausgehenden Kanten `out`.

Outline

Eine bequemere Schleife: `for-each`

`for`-each-Schleife

Die `for`-each-Schleife ist eine Schleifenvariante, die sich besonders gut eignet, um alle Elemente einer Sammlung zu verarbeiten.

```
for (<Variable> : <Sammlung>) {
   <Anweisung>;
   ...
}
```

- Die `for`-each-Schleife läuft durch die komplette angegebene Sammlung von vorne nach hinten durch. Kein Zähler nötig.
- Aber: Ändern der Richtung, Überspringen von Elementen, Entfernen oder Ändern von Elementen ist nicht möglich.

for-each vs. for

Reguläre for-Schleife:

```
List<String> myList = new ArrayList<String>();
// Werte der Listen-Elemente setzen

for ( int i=0; i<myList.size(); i++ ) {
   String element = myList.get(i);
   // tue etwas mit element
}
```

for-each-Schleife:

```
List<String> myList = new ArrayList<String>();
// Werte der Listen-Elemente setzen

for ( String element : myList ) {
   // tue etwas mit element
}
```

Outline

Pakete importieren

- Die Klasse `ArrayList` liegt in Java im Paket `java.util`.
- Java-Klassen werden in Paketen organisiert. Ein Paket ist eine Gruppe von logisch zusammenhängenden Klassen.

Pakete importieren

- Um Listen nutzen zu können, müssen wir die Klasse importieren:

  `import <Paketname>.<Klassenname>;`

```
import java.util.ArrayList;

public class Test {
  ...
}
```

- Mit dem Statement `import java.util.*;` werden *alle* Klassen des Pakets importiert.
- Importe stehen ganz am Anfang des Programms, also *vor* der Klassendeklaration.

Outline

Aufgabe: Suchen

Finden Sie die kleinste und die größte Zahl!

▶ 456	▶ 743	▶ 594	▶ 459	▶ 232
▶ 754	▶ 332	▶ 572	▶ 734	▶ 738
▶ 345	▶ 446	▶ 359	▶ 664	▶ 715
▶ 588	▶ 668	▶ 890	▶ 724	▶ 776
▶ 505	▶ 579	▶ 745	▶ 308	▶ 502
▶ 794	▶ 220	▶ 258	▶ 307	▶ 533
▶ 606	▶ 150	▶ 158	▶ 832	▶ 566

Aufgabe: Suchen

Finden Sie die kleinste und die größte Zahl!

- 108
- 114
- 133
- 155
- 185
- 205
- 209
- 223
- 276
- 302
- 308
- 338
- 341
- 354
- 372
- 390
- 414
- 435
- 458
- 470
- 536
- 550
- 602
- 616
- 622
- 642
- 649
- 706
- 759
- 790
- 806
- 824
- 848
- 892
- 895

Aufgabe: Suchen

Finden Sie die kleinste und die größte Zahl!

▶ 108	▶ 223	▶ 372	▶ 550	▶ 759
▶ 114	▶ 276	▶ 390	▶ 602	▶ 790
▶ 133	▶ 302	▶ 414	▶ 616	▶ 806
▶ 155	▶ 308	▶ 435	▶ 622	▶ 824
▶ 185	▶ 338	▶ 458	▶ 642	▶ 848
▶ 205	▶ 341	▶ 470	▶ 649	▶ 892
▶ 209	▶ 354	▶ 536	▶ 706	▶ 895

Suchen geht sehr viel einfacher, wenn Dinge sortiert sind!

Sortieren

Sortieren

Sortieren ist ein Prozess, der eine Anzahl an Elementen in absteigender oder aufsteigender Reihenfolge anordnet. Die Reihenfolge basiert auf einem Schlüssel, auf dem eine Ordnungsrelation definiert ist.

Beispiele:

- Sortiere Zahlen nach Größe.
- Sortiere Personen nach dem Schlüssel “Nachname” mit der Ordnungsrelation “alphabetisch”.
- Sortiere Personen nach dem Schlüssel “Geburtsdatum” mit der Ordnungsrelation “früher geboren”.

Aufgabe: Sortieren

Wie würden Sie folgende Zahlen sortieren?

$$2,\ 7,\ 4,\ 9,\ 1,\ 5$$

Überlegen Sie sich eine Schritt-für-Schritt-Anleitung.

SelectionSort

Anfangsstatus:

- Zu Beginn gelten alle Elemente als “unsortiert”.

Vorgehen:

- Solange es noch unsortierte Elemente gibt, wiederhole folgendes:
  - Suche unter allen unsortierten Elementen das kleinste aus.
  - Tausche dieses Element mit dem Element am linken Ende der Liste. Dieses Element gilt nun als “sortiert”.

Video hierzu:
`https://www.youtube.com/watch?v=y9B2oZV3iUI`

BubbleSort

Vorgehen:

- Solange es im letzten Durchlauf mindestens eine Vertauschung gab, wiederhole folgendes:
  - Beginne ganz links.
  - Wiederhole folgendes, solange es möglich ist:
    - Vergleiche das Element an der aktuellen Position mit dem Element rechts davon.
    - Falls das rechte Element kleiner ist, vertausche die beiden.
    - Gehe eine Position nach rechts.

Video hierzu:
`https://www.youtube.com/watch?v=qtXb0QnOceY`

Es gibt viele andere Sortieralgorithmen . . .

InsertionSort Sortieren durch Einfügen unsortierter Elemente in die bereits sortierte Liste

BinaryTreeSort Sortieren mithilfe eines binären Suchbaums.

MergeSort Sortieren durch Zerlegen in kleinere Listen und Zusamenfügen im Reißverschlussverfahren.

CocktailSort Bidirektionales Bubble-Sort.

Bucketsort Sortieren durch grobes Verteilen auf "Eimer" und Nachsortieren der Eimer.

Bogosort Sortieren durch Mischen und hoffen dass es irgendwann dabei zufälligerweise sortiert rauskommt.

. . .

- Vergleich: `https://www.toptal.com/developers/sorting-algorithms/`
- Mit Ton: `http://panthema.net/2013/sound-of-sorting/`

Outline

Zusammenfassung: Listen

- **Listen** (`java.util.ArrayList`) sind Datenstrukturen, die mehrere Variablen des gleichen Datentyps zusammenfassen.
- Listen haben eine variable Länge.
- Der Zugriff auf die einzelnen Array/Listen-Elemente erfolgt über einen **Index**. Indizes gehen von 0 bis Länge-1.
- Listen sind **generisch**, d. h. bei der Verwendung muss der Datentyp des Listeninhalts festgelegt werden.
- Mit listenähnlichen Klassen lassen sich auch andere Datenstrukturen implementieren.

Zusammenfassung: Sortieren

- Eine Klasse aus einem Paket wird **importiert** mit

  ```
  import <Paketname>.<Klassenname>;
  ```

- **Sortieren** ordnet Elementen in absteigender oder aufsteigender Reihenfolge nach einem **Schlüssel** an.
- Wir sortieren, weil es das Suchen leichter macht.
- Behandelte **Sortieralgorithmen**: SelectionSort, BubbleSort

Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Vielen Dank!

Frank Schweiner

E-Mail frank.schweiner@mint-kolleg.de
Telefon +49 (0) 711 685-84326
Fax —

Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Azenbergstr. 12
70174 Stuttgart