4AHIF

16.9.15

# Einführung in Java

Liefert Oracle für die versch. Plattformen und OS
-> Windows
-> Unix, Linux
-> PowerPC
-> MAC

Wieso ist Java so erfolgreich? → Plattformunabhängigkeit

(Markus)

```
~~import java.util.Scanner;~~
public class schachbrett{
 public static void main(String [] a)
 {
  int start = 1;
  int ende = 9;
 for(int x=1;x<9;x++)
  {
   for(int y=start;y<ende;y++)
   {
    if(y < 10)
    {
     System.out.print(" ");
    }
    System.out.print(y);
    System.out.print(" ");
   }
   System.out.println("");
   start++;
   ende++;
  }
 }
}
```

Modifizierer

Konvention: Klassenname gross!

Parameter

Einsprungpunkt beim Start

Zeilen

Spalten

Leerzeichen falls Einerstelle

Zahl ausgeben

Leerzeichen

Inkrement

Dekrement

c--

Korrektur Tannenbaum.java

18.9.15

Aliji

```
import java.util.Scanner;
public class Tannenbaum{

public static void main(String [] a)
{
 String stern = "*";
 String leer = "";
 int zahl = 0;
 Scanner s = new Scanner(System.in);
 System.out.println("wie viele Zeilen? :");
 int eingabe =  Integer.parseInt(s.nextLine());
 zahl = eingabe;
 for(int i = 0; i < eingabe; i++)
 {
  for(int x = 0; x < zahl; x++)
  {
   leer = leer + " ";
  }
  System.out.println(leer + stern);
  stern = stern + "**";
  zahl--;
  leer = "";
 }
 if(eingabe != 0 && eingabe < 20)
 {
  for(int y = 0; y < stern.length()/2;y++)
  {
   System.out.print(" ");
  }
  System.out.print("I");
 }
```

String-Parameter
↳ Rückgabewert der Methode zählt Zeilen

Abstand erstellen

spart sich Schleife

Stamm berechnen

Lesbarkeit ⇓ Funktion OK

Statistik (Einschub #)

23.9.15

Ergebnis: Zufallszahlenverteilung

25.9.15

je höher die Zahl der Versuche ⇒ umso gleichverteilter ist das Ergebnis

Normalverteilung

Wie können andersartige Häufigkeiten erzeugen?

Tabelle

| | |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 6 |

Zahl von 1 - 100 gleichverteilt

---

Korrekter Statistik

1) Version ohne Arrays

2) Version mit Arrays

 Koczwara

```
import java.util.Scanner;
import java.util.Random;

public class StatistikOLD{
```

Memberfeld: static Random r = ....

```
        public static void main(String [] a)
        {
                System.out.println("Statistik");

                Scanner s = new Scanner(System.in);
                System.out.print("Geben sie Anzahl der Zahlen ein: ");
                int eingabe = s.nextInt();
                System.out.println();

                int zahl0=0;
                int zahl1=0;
                int zahl2=0;
                int zahl3=0;
                int zahl4=0;
                int zahl5=0;
                int zahl6=0;
                int zahl7=0;
                int zahl8=0;
                int zahl9=0;



                int maxVal = 0;
                int durchschnitt;

                for(int i = 0; i < eingabe; i++)
                {
                        switch(getZufall())
                        {
                                case 0: zahl0++;
                                                break;

                                case 1: zahl1++;
                                                break;

                                case 2: zahl2++;
                                                break;

                                case 3: zahl3++;
                                                break;

                                case 4: zahl4++;
                                                break;

                                case 5: zahl5++;
                                                break;

                                case 6: zahl6++;
                                                break;

                                case 7: zahl7++;
                                                break;

                                case 8: zahl8++;
                                                break;

                                case 9: zahl9++;
                                                break;
                        }

                        if(maxVal < zahl9)
                                maxVal = zahl9;

                        if(maxVal < zahl8)
```

```
                        maxVal = zahl8;

                if(maxVal < zahl7)
                        maxVal = zahl7;

                if(maxVal < zahl6)
                        maxVal = zahl6;

                if(maxVal < zahl5)
                        maxVal = zahl5;

                if(maxVal < zahl4)
                        maxVal = zahl4;

                if(maxVal < zahl3)
                        maxVal = zahl3;

                if(maxVal < zahl2)
                        maxVal = zahl2;

                if(maxVal < zahl1)
                        maxVal = zahl1;

                if(maxVal < zahl0)
                        maxVal = zahl0;
        }


        durchschnitt = maxVal / 50;
        if(durchschnitt == 0)
        {
        durchschnitt = 1;
        }
```

falsche Normierung

```
        System.out.print("Legende: ein X entspricht " + (durchschnitt) +
" Treffern");
        System.out.println();

        System.out.print("0: ");
        for(int y = 0; y < (zahl0/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("1: ");
        for(int y = 0; y < (zahl1/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("2: ");
        for(int y = 0; y < (zahl2/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("3: ");
        for(int y = 0; y < (zahl3/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("4: ");
        for(int y = 0; y < (zahl4/durchschnitt); y++)
        {
```

```
                System.out.print("X");
        }
        System.out.println();

        System.out.print("5: ");
        for(int y = 0; y < (zahl5/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("6: ");
        for(int y = 0; y < (zahl6/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("7: ");
        for(int y = 0; y < (zahl7/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("8: ");
        for(int y = 0; y < (zahl8/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();

        System.out.print("9: ");
        for(int y = 0; y < (zahl9/durchschnitt); y++)
        {
                System.out.print("X");
        }
        System.out.println();
    }
    static int getZufall()
    {
        Random random = new Random();
        int randomNum = random.nextInt(10);
        return randomNum;
    }
}
```

jedes mal neues Objekt sollte man vermeiden

sollte ins Member feld

besser mit Arrays

StatistikArr.java

Bauer

```
import java.util.Scanner;
public class StatistikArr{
        public static void main(String [] a){
                Scanner s = new Scanner(System.in);
                System.out.print("Anzahl der Zufallszahlen: ");
                int max = 0, rndm, liste[] = new int[10], anzahl =
Integer.parseInt(s.nextLine());//Deklaration und Eingabe
                System.out.println("");
                for(int i = 1; i <= anzahl; i++){//Generiert die Zufallszahlen
und zählt sie
                        rndm = getZufall();
                        liste[rndm]++;
                }
                if(groestezahl(liste) < 50){
                        max = 50;
                }
                else{
                        max = groestezahl(liste);
                }
                for(int j=0; j <=9; j++){//Geht den index des Arrays durch(0-9)
                        System.out.print(j + ": ");
                        for(int z=1; z <=liste[j]/(max/50); z++){//Schaut, dass
es nie über 50 "X" werden und gibt sie dementsprechend aus.
                                System.out.print("X");
                        }
                        System.out.println("");
                }
                System.out.println("Legende: Ein X entspricht " + max/50 + "
Treffer(n)");
        }
        public static int getZufall(){//Gibt eine Zufällige Zahl zwischen 0-9
zurück
                int randomNum = 0 + (int)(Math.random()*10);
                return randomNum;
        }
        public static int groestezahl(int array[]){//Gibt die größte Zahl aus
dem Array zurück
                int max = 0;
                for(int i = 0; i<=array.length - 1; i++){
                        if(array[i] > max){
                                max = array[i];
                        }
                }
                return max;
        }
}//Oj is DMG.
```

# Zahlenraten korrektur

Zahlenraten.java

koczwara

```
import java.util.Scanner;
import java.util.Random;
public class Zahlenraten{ //Alec
        static Random random = new Random();
        static int getZufall()
        {
                int randomNum = random.nextInt(1000);
                return randomNum++;
        }
        public static void main(String [] a)
        {
                Scanner s = new Scanner(System.in);
                int eingabe, antwort, randNum, versuch=0;
                System.out.println("\nZahlenraten");
                System.out.println("Erraten Sie eine Zahl zwischen 1 und
1000.");
                do{ //Spielschleife
                        randNum = getZufall();
                        do{ //Versuchschleife
                                versuch++;
                                System.out.println("\nVersuch: " + versuch);
                                System.out.print("Geben sie ein: ");
                                eingabe = s.nextInt();
                                if(eingabe > randNum){
                                        System.out.println("Die Zahl ist zu
hoch. \n");
                                }
                                else if(eingabe < randNum){
                                        System.out.println("Die Zahl ist zu
niedrig. \n");
                                }
                        }while(eingabe != randNum && versuch < 10);
                        if(eingabe == randNum){
                                System.out.print("Die Zahl ist richtig! \n");
                        }
                        versuch=0;
                        System.out.print("\nWollen Sie nochmal spielen? (Ja = 1
/ Nein = 2) \n");
                        antwort = s.nextInt();
                }while(antwort == 1);
        }
}
```

prelukrement erforderlich oder (a+1)

++

0 – 999

1 – 1000

Vorgabe wortwörtlich umgesetzt!

Zeilenumbruch

## IDE verwenden: eclipse

USER

Workspace

Dateiverzeichnis

ordner verwaltet durch eclipse

. metadata

01 Einführung

02 Arrays

Sourcen und binaries

"Finger weg!"

30.9.15

# Arrays

Übungen zu Arrays AB!

1.10.15

## Informationen zu Arrays

**Deklaration:** char[] a; Point[] p;

**Erzeugen von Arrays:** p = new Point[5]; → 5 Elemente

**Initialisieren**

über Index jedes Element zuweisen! (z.B. Schleife)

p[0] = new Point();

⋮

p[4] = new Point();

---

## Mehrdimensionale Arrays

int t[][] = new int[3][4];

t[1][1] = 5;

↳ initialisieren mit 2 geschachtelte Schleife

## Array: schnelles Initialisieren (Testprogs)

int feld[] = {1, 2, 3, 4}; →

| Index 0 | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

int f2[][] = {{1, 2, 3}, {2, 3, 4}};

| 1. Index \ 2. Index | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 2 | 3 | 4 |

f2[1][2];

int l1 = f2[1].length ergibt jeweilige Z-Länge

f2.length ergibt Anzahl der Zeilen

## Array Grenzen

geg: array a

Länge: int l = a.length; ← Parameter "member"

~~Methode~~ ~~keine~~ -> ~~length()~~

int b = new int [3];

b[3] = 5;

Programmabbruch

array OutofBoundException

---

## Ändern von Array-Größe

int [] c = {3, 7, 2};

~~c.add(5)~~

nur neu erstellen! → größer: int c1[] = new int[4]

(+) Element weise kopieren!

→ kleiner

---

## Arrays kopieren (seit Java 6)

neue Länge

java.util.Arrays → public static T[] copyOf(T[] org, int l)

sehr alt: (Java 1.0)

System.arraycopy(Quelle Länge Ziel Länge)

---

## verbesserte Schleife

array

for (int element : array) {*}

iteriert alle Elemente von array

* System.out.println(element); ⇒ 1 2 3 4

## Strings kopieren

String s[] = { "H", "a", "L", "L", "o" }; 5 Elemente

Anmerkung:

die einzelne String sind nicht als Text im Speicherplatz dort stehen auch Referenzen.

Seine Referenz

int bei primitives

a = 3 (Zahlen)

b = a

a = 4

b => 3

String s1 = s;

↑ kopiere hier nur Referenz

s[4] = "i";

s1[4] ist jetzt auch (i)

≙ gleicher Speicherplatz

## shallow copy

geg: s String mit 5 Elementen

String [] s1 = Arrays.copyOf(s, 5);

# deep Copy

```
public class Person {
        String name;
        String Vorname;
        Note [] n;
    ...
}
```

```
class Note {
    int note;
    double schnitt;
    ...
}
```

```
Person p [] = { new Person(... new Note ....) }
```

p1 = p; p2 = Arrays.copyOf(p, 2)

deepcopy: es müssen die Objekte, die hinter den kopierten Verweisen/Referenzen stehen auch dupliziert werden.

alle !

Gefahr => unendlich

---

Ablaufsteuerung hardy ✓ Modifier Private public

Ablaufsteuerung hardy mit Konstruktoren ✓

HÜ für Mi 7.10. Reihenfolge !

bitte an nicht Anwesende weitergeben!

public int tuWas (String s) { .... };

im Klassendiagramm: tuWas (String s) : int

UML-Klassendiagramm

## Aufgabe: Übungen zu Polymorphie in Java

05 Projekt: Polymorphie

Gegeben sei in Hauptropgramm PolyHaupt.

```
public class PolyHaupt {

      public static void main(String[] args) {

            PolyPerson pp0 = new PolyPerson();
            PolyPerson pp1 = new PolyVater();
            PolyPerson pp2 = new PolyKind();
            PolyPerson pp3 = new PolyEnkel();

            pp0.sagMeinung();
            pp1.sagMeinung();
            pp2.sagMeinung();
            pp3.sagMeinung();

            System.out.println("Ende der Vorstellung,
jetzt experimentieren wir");

            pp0  =  pp2;
            pp0.sagMeinung();

            PolyEnkel ppx;
            ppx  = (PolyEnkel) pp3 //Zuseisung pp3
            ppx.sagMeinung();

      }
}
```

members = Variablen

Methoden (Konstruktoren)

Klassenname

Vererbung "extends"

↳ Fehler!

pp3 ist Polyperson und weiss nicht dass es "Enkel" ist

**Aufgabe1:** Codiere folgende Klassen!
Die jeweiligen Methoden `sagMeinung():void` geben auf der Konsole folgende Texte aus:

- PolyPerson: „Eine Person ist neutral“
- PolyVater: „Ich bin ein Vater“
- PolyKind: „Ich bin ein Kind“
- PolyEnkel: „Ich bin ein Enkel“

Beobachtung:

Konsolenausgabe:

```
Eine Person ist neutral
Ich bin ein Vater
Ich bin ein Kind
Ich bin ein Enkel
Ende der Vorstellung, jetzt experimentieren
wir
Ich bin ein Kind
Ich bin ein Enkel
```

Abhilfe: (PolyEnkel) Typecast

Erkläre den gezeigten Codeabschnitt und überlege die damit verbundene Problematik

```
PolyEnkel ppx;
ppx = (PolyEnkel) pp3 //Zuseisung pp3
ppx.sagMeinung();
```

erzwingt Typwandlung

Aufgabe 2:

Erweitern Sie die PolyHaupt mit einer Methode, die die Nutzung eines polymorphen Argumentes demonstrieren kann.

Zeigen Sie auch die Anwendung des instanceof Operators in diesem Beispiel.

Aufgabe 3:

Erstellen Sie eine Klasse Statistik ( als Hauptprogramm ) die eine vararg-Methode enthält zum Errechnen des Durchschnitts von ganzen Zahlen.

Aufgabe 4: freiwillige Zusatzaufgabe

Kopieren sie die Klasse MeinDatum aus dem Vorgängerprojekt. Überschreiben Sie die geerbte equals(), die hash() und die toString() Methode

(Hinweis, der Rückgabewert int der hash() Methode sollte in geeigneter Weise gleiche Datums mit einer gleichen int Zahl signalisieren)

zu 1) Polymorphie ermöglicht Objekte aus einer Vererbungshierarchie auf einer Variable der Oberklasse zu verwenden. Die Methoden funktionieren bleiben erhalten

zu 2) Polymorphes Argument an Beispiel

```
public void tuWas (PolyPerson p)
{ p.sagMeinung();
}
```

Man kann abgeleitete Elemente übergeben: p als PolyKind oder PolyEnkel

Voraussetzung: Vererbungshierarchie

instanceof : Test ob Objekt in einer Vererbungshierarchie stehen!

p instanceof Polyperson / true / false ✓

---

zu A3 var Arg()

```
public void tuWas(String... s){ }
```

1) Aufruf: tuWas("1"); ⇒ s[] → "1" Länge 1 → 1

2) Aufruf: tuWas("1", "2", "3"); ⇒ s[] der Länge 3 → 1 | 2 | 3

3) Aufruf: tuWas() ⇒ s[] → Länge 0

---

05 Polymorphie

9.10.15

## Wrapperklassen

Primitive Variablen in Java:

- int (4 Byte), double (8 Byte), boolean (1 Byte), char (2 Byte, unicode 16), float (4 Byte)
- short (2 Byte), byte (1 Byte), long (8 Byte)

~~String~~

String s = "X"

int a = 7;

Integer i = new Integer("2");

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

```
int i = 5;
Integer iw = Integer.valueOf(i)
```

Rückgabewert ist Objekt

**Boxing**

```
int i2 = iw.intValue();
```

Rückgabewert primitiv

**Unboxing**

---

```
Character aw = 'a';
```

✓ **Autoboxing**

```
Integer iw2 = 3;
```

✓ (seit Java 5)

```
int z = iw2;
```

✓ **Autounboxing**

---

Übung zu Wrapperklassen:

System.currentTimeMillis() → Long

16.10.15

bei Schleife für Zeitmessung

```
Integer iw = i;
Integer iw_copy = iw;
```

(Autoboxing)

↳ Compiler optimiert ggf.

soll vermieden werden

## Ergebnisse

Zeit /ms

| Anzahl | primitive Kopi | Boxing | Autoboxing | |
|---|---|---|---|---|
| 1.000.000 | 15 | 16 | 15 | surface 1 |
| 1.000.000.000 | 2 | 6 | 1100 | Asus (i5) |
| 1.000.000 | 3 | 4 | 9 | |
| 342.352.352 | 9 | 378 | 377 | |
| 100.000.000. | 44 | 495 | 345 | |

---

## Aufzählungstypen ≙ Enumeration "Enum's"

```
public class WoTa {

public final int MONTAG = 1;
      "     "    "  DIENSTAG = 2
      "     "    "
      ⋮
}
```

int a = WoTa.MONTAG
int b = WoTa.DIENSTAG
int c = a + b ; ✓
↓
3 ⇝ M.Hwood

RGB Wert ≙ 3 Bytes

Verwendung bei Farben
Color.RED

↳ oder besser: enum Type

16.10.15

# Einschub: statische Variable

Statische Variablen mit dem Klassennamen ansprechen!

---

Übung Aufgabe 2 — 3 Arten für Aufzählungstypen ✓

HÜ ⇒ Mi 21.10 : • AB: erw. Klassenfunktionen $\boxed{A1}$ Objektzähler

⇒ • Alternative für switch(Farbnr) ⇝ Farbname: String

---

⇒ Mi ⇒ statischer Import

⇒ abstr. Klasse + Interfaces

21.10.15

✓ ⇒ zu HÜ: statt switch(farbe) zur Zuordnung der Farbnamen können wir ein Array: String verwenden

Farbnr ⇒ index-0 ⇝ h: rot, gelb, schwarz ⇝ String

⇒ statischer Import — import static java.lang.Math.sqrt;

✓ ↯ sparsam verwenden da Namensraum eingeschränkt wird ⇝ sqrt(1.1) aufrufen, als wenn es eigene Klassenfunktion wäre

# Abstrakte Klassen

A.-Klasse legt nur fest welche Methoden in abgeleiteten Klassen ausprogrammiert werden müssen

```
abstract class A {
    public abstract void tuWas();
    public void tunochWas(){};
}

class B extends A {
    void tuWas() { ..... };
```

← ohne Rumpf "Body" => abstract

tut nix ist aber implementiert

22.10.15

## Interfaces

```
public class C implements I {
    public void tuWas() {....},
}
```

erzwingt

- es können mehrere Interfaces implementiert werden!

≠ abstr. Klassen nur 1x extends

```
interface I {
    public void tuWas();
    public static int a;
    public final static B = 3;
    public void tunochWas(){....};
}
```

init fehlt ✓

nicht erlaubt

- Sammlung von Methodenköpfen!
- ~~ausprogrammierte Methoden~~ nicht erlaubt
- Variablen nicht erlaubt
- Konstanten erlaubt + (Compiler ergänzt { public, final

# Exceptions – Ausnahmebehandlung in Java

30.10.15

try ✓
catch ✓
throw ✓
finally ✓
Exception ✓
throws ✓

o mit Objekt belegt

```
try {
  o.tuWas();
  xxxx
}catch(Exception e ){
  // code für den Fehlerfall
  e.irgendwas();
} finally {
  ⋮
  // wird immer ausgeführt
  ⋮
}
```

(ob Fehler oder nicht)

---

**throws** Exception : Fehler an die übergeordnete Einheit (Aufrufer) weitergeleitet.

- letztlich bis zur Klasse, die die main(...) enthält → Betriebssystem muss Fehler bearbeiten ⇝ JVM Programmabruch

HÜ: 4; 4.11; A3 + Theorie von Exceptions

04.11.15

# Zusicherungen / Assertions

sollen für den Programmierer "zusichern", dass bestimmte Bedingungen im Code erfüllt sind.

true/false

als String (auch Rückgabewert evaluiert)

assert < Bedingung > : "Fehlermeldung" ;

und $\boxed{-ea}$ als VM argument nicht vergessen!

$\boxed{-enable\ assertions}$

java -ea Mein Programm

Keine HÜ! → Theorie Collections!

08 Collection

5.11.15

Arrays haben viele Nachteile !

- Element hinzufügen bedeutet neues Array machen + altes Array kopieren

# Interface Collection

Sammlung dynamischer Datenstrukturen – passen ihre Größe an die Anzahl der Daten an

Doublettenlos
"Menge" es gibt weder index noch key
extends (erbt)
<interface> Collection
+ add
+ remove
+ contains
+ size
+ isempty
+ interator
erbt
<interface> Map
+ get
+ put
+ value
+ keyset
+ entrySet
Assoziativer Speicher
key, Value
1 Paare
Dictionary
<interface> Set
extends
implementiert
<interface> SortedSet
implements
<interface> List
+ get
+ set
+ indexOf
+ size
+ isempty
+ interator
<interface> SortedMap
HashTable
key, Value
TreeMap
Properties
key: String
Value: String
HashSet
TreeSet
ausprogrammiert
konkrete Klasse
impl.
impl.
ArrayList
LinkedList
Verkettete Liste
index wahlfreier Zugriff
konkret
HashMap
LinkedHashMap

# List /Set/Map

- Set
  - Repräsentiert eine doublettenlose Menge von Elementen auf die mit Mengenoperationen zugegriffen wird
- List
  - Repräsentiert eine Liste von Elementen beliebigen Typs auf die wahlfrei od sequentiell zugegriffen werden kann
- Map
  - Repräsentiert eine Menge von Objektpaaren, wobei jeweils ein Objekt ein doublettenlosen Schlüssel zum Zugriff auf das dazugehörige andere Objek darstellt (Key – Value Paar)

Übung: AB: A1 + A2 Set + List Beispiel !

11.11 HÜ: dyn. Listen kennen → A3 "Entscheidungsträffe

**Aufgabe: Übungen zum neuen Collections Framework in Java**
Projekt: 08Collections

Aufgabe1: Erstellen Sie die Klasse `SetBeispiel` (soll auch ausführbar sein)

Erzeugen Sie ein Set (HashSet) und fügen sie verschiedene Objekte hinzu (z.B. Strings und div. Wrapperobjekte).

Beobachten Sie die Ausgabe des Sets auf der Konsole. Was passiert, wenn Duplikate hinzugefügt werden?

Aufgabe2: Erstellen Sie die Klasse `ListBeispiel` (soll auch ausführbar sein)

Erzeugen Sie ein List-Objekt (ArrayList) und fügen sie verschiedene Objekte hinzu. (z.B. Strings und div. Wrapperobjekte).

Beobachten Sie die Ausgabe. Was passiert, wenn Duplikate hinzugefügt werden?

**Entscheidungshilfe – welche Klassen sind optimal?**

Alphabetisch

1. Ist eine Sequenz, also eine feste Ordnung gefordert? TreeSet
2. Soll es einen schnellen Zugriff über einen Index geben? ArrayList
3. Werden oft am Ende und Anfang Elemente eingefügt? → LinkedList
4. Wenn eine Reihenfolge der Elemente uninteressant ist, aber schnell entschieden werden soll, ob ein Element Teil einer Menge ist, erweist sich HashSet.
5. Sollen Elemente nur einmal vorkommen und immer sortiert bleiben? → TreeSet
6. Muss es eine Assoziation zwischen Schlüssel und Elemente geben. → HashMap

# nützliche Implementierungen (dyn. Listen)

11.11.15

| | Hash Tabelle | Veränderbares Array | B-Baum "sortiert" | Verkettete Listen | Verkettete Liste + Hash Tabelle |
|---|---|---|---|---|---|
| Set | HashSet | | TreeSet | | LinkedHashSet |
| List | | ArrayList | | LinkedList | |
| Deque (double ended queue) | | ArrayDeque | | LinkedList | |
| Map | HashMap | | TreeMap | | LinkedHashMap |

## Legatklasse (Legacy classes) Kompatibilität zu Java 1.0

- Vector implementiert List Interface
- Stack ist Erweiterung von Vector mit push(), pop(), peek().
- Hashtable implementiert Map
- Properties Erweiterung von Hashtable ⇝ aber nur String als Key/Value Paare

Hü ⇝ 12.11. Collections AB2 : A1 + A2