Informatik II

Übungsstunde 2

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich

Problem: Uebungsabgabe

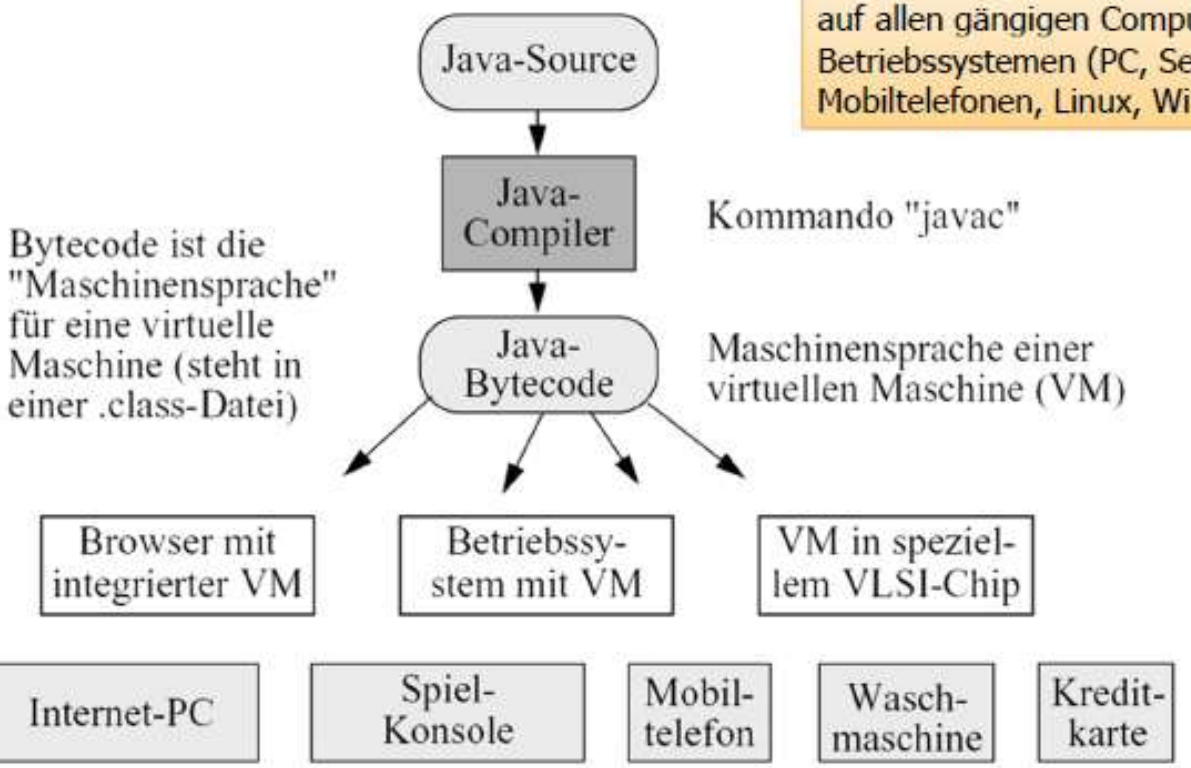
- Ueberprueft, dass das Archiv wirklich alles Relevante enthaelt!
- Serie 1:
  - Code (.java Dateien, .class sind nicht ausreichend!)
  - Tests
  - Javadoc (im /doc Folder innerhalb des Eclipse Projekts
  - Theoretische Uebungen
- Als .zip (oder .tar.gz), bitte kein .rar

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Baeume, Rekursion, Sortieren
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Plattformunabhängigkeit durch Bytecode-Interpretation

Ein Java-Programm läuft (prinzipiell) auf allen gängigen Computern und Betriebssystemen (PC, Server, Mobiltelefonen, Linux, Windows...)



Einfache Datentypen in Java

Bildnachweis: Christian Ullenboom: *Java ist auch eine Insel*, Galileo Computing, 8. Auflage, 2009, ISBN 3-8362-1371-4

Einfache Datentypen in Java (II)

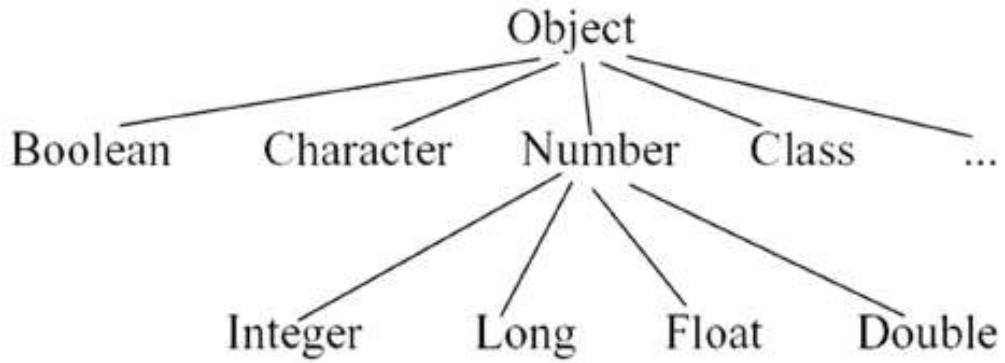
Datentyp	Größe[a]	Wrapper-Klasse	Wertebereich	Beschreibung
boolean	1 Bit	java.lang.Boolean	true / false	Boolescher Wahrheitswert
char	16 Bit	java.lang.Character	U+0000 ... U+FFFF	Unicode-Zeichen (z. B. `'A'` oder `'\uC3A4'`)
byte	8 Bit	java.lang.Byte	−128 ... +127	Zweierkomplement-Wert
short	16 Bit	java.lang.Short	−32.768 ... +32.767	Zweierkomplement-Wert
int	32 Bit	java.lang.Integer	−2.147.483.648 ... +2.147.483.647	Zweierkomplement-Wert
long	64 Bit	java.lang.Long	−9.223.372.036.854.775.808 ... +9.223.372.036.854.775.807	Zweierkomplement-Wert
float	32 Bit	java.lang.Float	±1,4E−45 ... ±3,4E+38	Gleitkommazahl (IEEE 754)
double	64 Bit	java.lang.Double	±4,9E−324 ... ±1,7E+308	Gleitkommazahl doppelter Genauigkeit (IEEE 754)

[a] „Größe" gibt den minimalen Speicherverbrauch an.

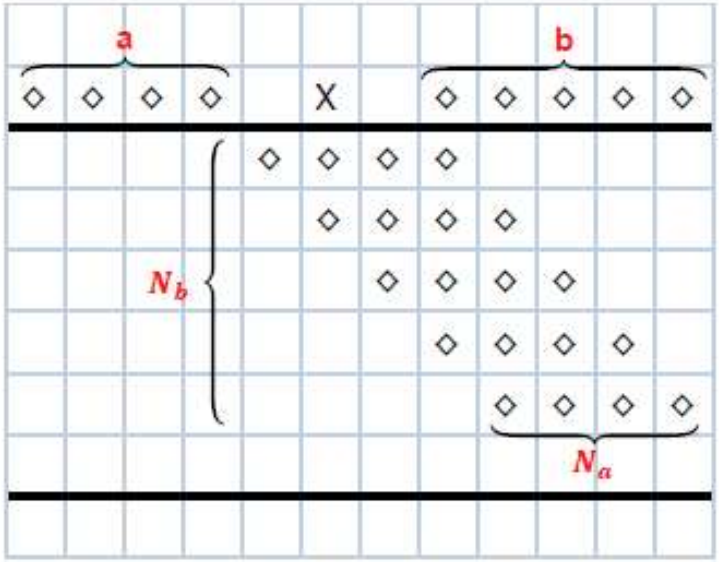
Bildnachweis: http://de.wikipedia.org/wiki/Java-Syntax

Hüllenklassen

- Einfache Datentypen (int, float,...) sind a priori keine echten Objekte (zu grosser Aufwand → ineffizient!)
- Für diese gibt es sogenannte Hüllenklassen
- Objekte davon können überall dort verwendet werden, wo eine Objektreferenz verlangt wird



Aufwands- / Qualitätsvergleich (II)



$N_a = 1 + \lfloor \log_{10} a \rfloor \leq 1 + \log_{10} a$

$N_b = 1 + \lfloor \log_{10} b \rfloor \leq 1 + \log_{10} b$

$\cong 0.3 \log_2 b$

- Etwa $N_a \times N_b$ kleine Multiplikationen und fast ebensoviele Additionen, also insgesamt $\approx 2(\log_{10} a)(\log_{10} b)$ elementare Operationen

Wann ist $5 \log_2 b$ besser als $2 (\log_{10} a) (\log_{10} b)$?

- Sind die multiplikativen Konstanten 5 bzw. 2 exakt / wichtig?
- Hängt wesentlich von a ab: $2 (\log_{10} a) (\log_{10} b)$ kann (bei festem b und grösserem a) über alle Grenzen wachsen!

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Baeume, Rekursion, Sortieren
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Übung 1 (Nachbesprechung)

- Oft nicht alles enthalten

- 2.b. hatte immerhin eine Gruppe richtig.
  - Sehen wir's uns kurz an...
- 2.c. hat niemand richtig. Also: Halb so schlimm ☺
  - Sehen wir's uns kurz an...
- Ab jetzt: Deutlich weniger theoretisches Zeug, mehr Programme

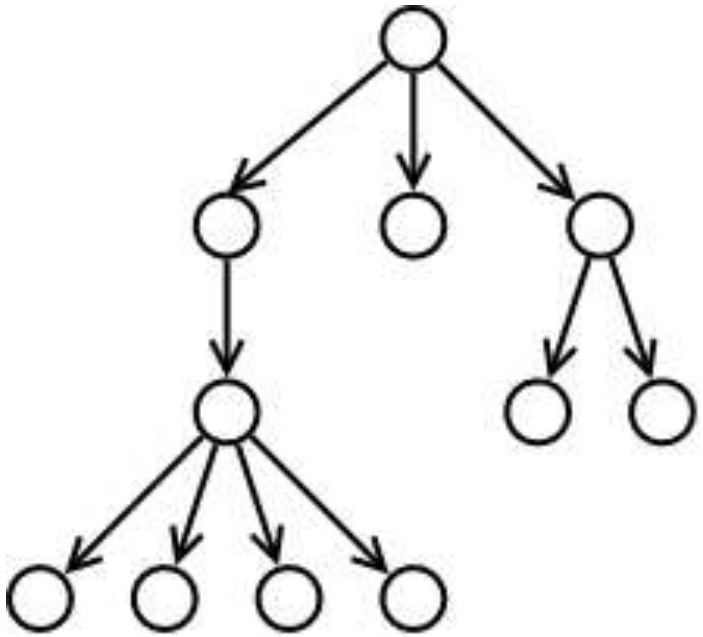
Übung 2

- 1. Wurzelbaeume
  - Trennung von Struktur und Darstellung
  - Klammerdarstellung
  - Darstellung in eingerueckter Form
- 2. Rekursives Sortieren
  - Ausgabe von Objekten mittels toString()
  - Rekursiver Sortieralgorithmus
- 3. Binaerbaeume als Arrays
  - Wichtigste Sache: checkTree

Übung 2 – Aufgaben 1 und 3

- Baeume in der Informatik...

vs.



Images: http://kitabundsunnah.wordpress.com/
http://www.cs.lmu.edu/courses

Übung 2 – Aufgaben 1 und 3

- Verschiedene Arten/Typen von Baeumen (mit Demos!)
  - Binaerbaum: *Jeder Knoten hat maximal 2 Kinder*
  - Binaerer/Triaerer Suchbaum: *Geordnetes Speichern von Knoten*
  - Selbstbalancierender Suchbaum: *Verhindert Baum-Degeneration*
    z.B. Rot-Schwarz Baum
  - Trie (von «Re**trie**val»): *Nicht Knoteninhalt, sondern Knotenposition ist wichtig*
    z.B. Suffixbaum
    *d.h.: **Kanten** enthalten Information!*

Übung 2 – Aufgabe 3

- Speichern von Binaerbaeumen als Arrays

  `[ 2 4 1 6 _ 3 4 9 5 4 ]`

  Geht das auch mit allgemeinen (=nicht-binaeren) Baeumen?

- Ueberpruefung der «Baumeigenschaft» eines Arrays:

  `[ 2 _ 1 6 _ 3 4 9 5 4 ]` ist kein Baum!

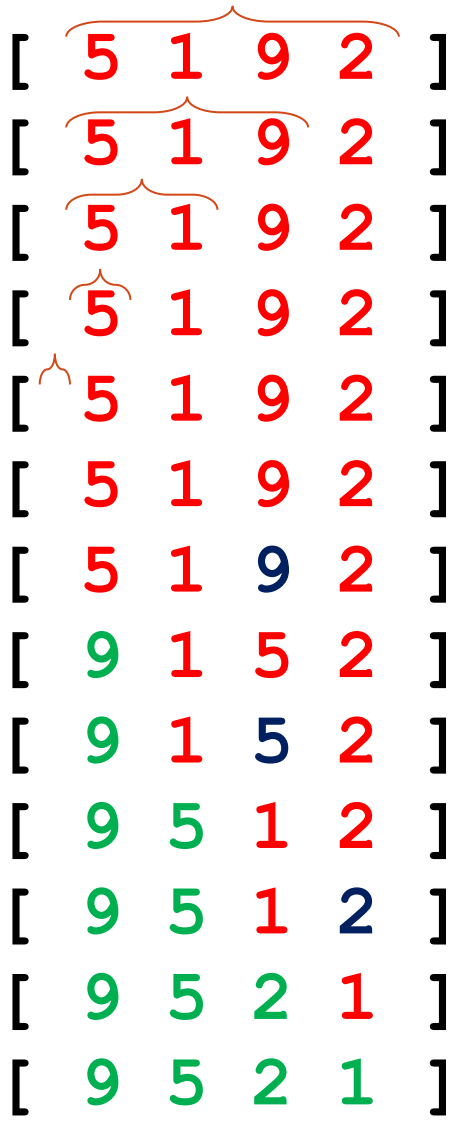
  Warum?
  Wie stellt man das programmatisch fest?

Übung 2 – Aufgabe 2

- Kernidee der Rekursion: Reduzieren einer Probleminstanz auf eine *kleinere* Probleminstanz.
- Gegeben: Liste mit n Elementen

```
Um eine Teilliste mit i Elementen absteigend zu sortieren,
brauche ich nur...
        ... die ersten (i - 1) Elemente absteigend sortieren
        ... das groesste Element im Rest der Liste suchen
        ... und an die erste Stelle des Restes der Liste setzen
```

- Die leere Liste ist selbstverstaendlich schon sortiert... ;-)




```
recursiveSort(4)
    recursiveSort(3)
        recursiveSort(2)
            recursiveSort(1)
                recursiveSort(0)
                Ist sortiert!
            9 <- findLargest(0,3)
            Swap
        5 <- findLargest(1,3)
        Swap
    2 <- findLargest(2,3)
    Swap
Kein swap mehr noetig...
```

→ Liste absteigend sortiert!

Koennten wir die Rekursion denn schon bei *recursiveSort(1)* abbrechen?

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Baeume, Rekursion, Sortieren
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Projekt: Baeume und Rekursion

- Idee: Um eine Liste von Zahlen zu sortieren, koennten wir sie in einen binaeren Suchbaum einfuegen und dann in-order auslesen.
- Archiv mit Programmgeruest habt ihr im Posteingang...
- Eingefuehrte Java Konzepte:
  - Interfaces
  - Iteratoren
  - Templating
  - java.util.List, java.util.ArrayList

Informatik II

Übungsstunde 2

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich