II.4.3 Modularität und Pakete

Mittwoch, 14. Dezember 2016 09:30

- Software sollte modular aufgebaut sein.
Module können von verschiedenen Personen entwickelt + gewartet werden.
- In Java werden Module durch Pakete realisiert (package).
- package: Zusammenfassung von Klassen und Interfaces, die über mehrere Dateien verteilt sein können.
- Bsp: Paket listen, enthält Klassen Element und Liste.
⇒ Jede Datei muss mit package listen; beginnen.
- Java verlangt, dass die Paket-Struktur der Struktur der Verzeichnisse entspricht.
⇒ Element.java und Liste.java müssen in einem Verzeichnis listen stehen.
(Paketname = Verzeichnisname)

d.h. Das Arbeitsverzeichnis muss ein Unterverzeichnis "listen" haben, das die Dateien Liste.java und Element.java enthält

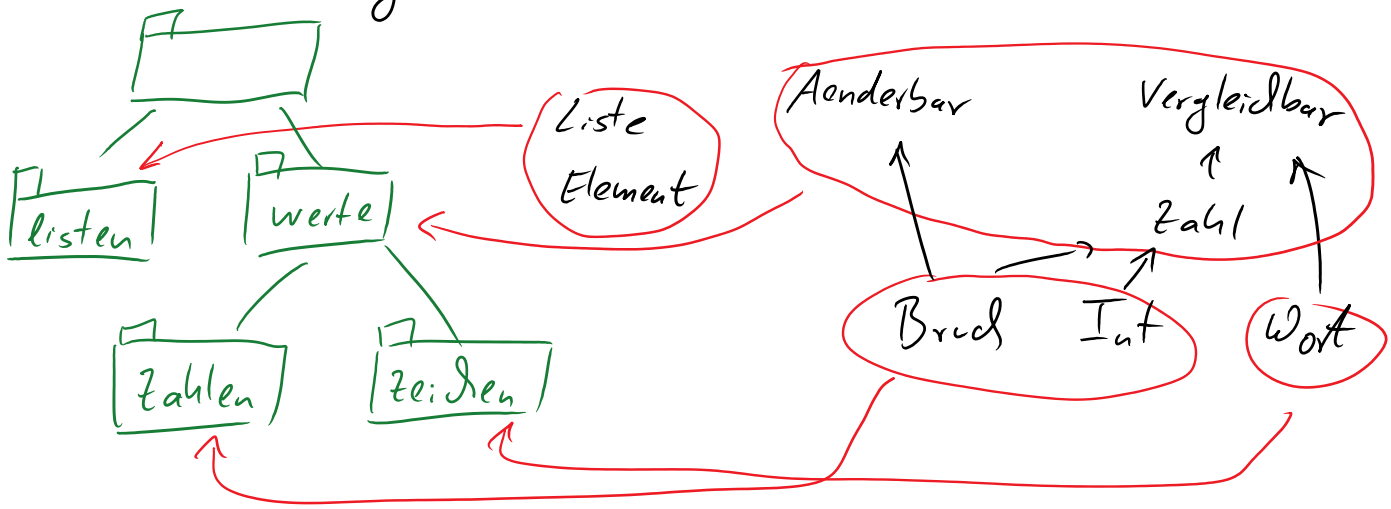
- Zugriff auf Klassen aus anderen Paketen: import

← erst package ....
dann import ...

- public: überall zugreifbar
  kein Schlüsselwort: Zugriff nur im eigenen Paket
  protected: Zugriff im eigenen Paket u. in Unterklassen
  private: Zugriff in eigener Klasse
- Wenn kein package am Anfang: Standardpaket ohne bestimmten Namen.
- Bsp: Compilierung von Test.java im Arbeitsverzeichnis bewirkt Compilierg. von Liste.java, Element.java im Unterverzeichnis listen.
  Alternativ: javac listen/Liste.java in Arbeitsverzeichnis
- Paket exportiert alle public- und protected-Komponenten.
- Pakethierarchie
  - entspricht Hierarchie von Ober- und Unterverzeichnissen
    Bsp: Neben Verzeichnis "listen" könnte es weiteres Verzeichnis "werte". Dieses enthält 2 weitere Unterverzeichnisse "werte.zahlen" und "werte.zeichen"
  - Pakete sind logisch unabhängig: Man benötigt "import", um vom Ober- auf Unterpaket oder umgekehrt zuzu-

greifen.

- Kein Zusammenhang zwischen Paket- und Klassenhierarchie:



Zusammenfassung

- Pakete fassen logisch zusammenhängende Klassen + Interfaces zusammen.
- Änderungsaufwand sollte möglichst nur 1 Paket betreffen
- Zyklische Importe => schlechte Modularisierung