II.4.3 Modularität und Pakete

Freitag, 1. Dezember 2017 09:30

- Software sollte modular aufgebaut sein.
- Verschiedene Teile sollen v. unterschiedl. Personen entwickelt u. gewartet werden.
- Änderungen sollten mögl. auf einzelne Teile begrenzt bleiben.
- Java:
  - Pakete (package) : Sammlungen v. zugeh. Klassen + Interfaces
  - Module (module) : Sammlungen v. Paketen

Bsp: Element und Liste sollen in gleichem Paket listen sein.

- Paketstruktur ≙ Dateistruktur
  im Bsp: Die Dateien Element.java und Liste.java müssen in Verzeichnis listen sein.
- Zugriff auf Klassen aus anderen Paketen:
  - Nur public und protected-Teile sind in anderen Paketen sichtbar.
- Zugriff auf Klasse aus anderem

Paket durch explizite Angabe des Paketnamens.

(Im Bsp stehen die Test-Klassen in einem anderen Paket als Listen.)

- Wenn man kein Paket vereinbart (kein package ... ;), dann gehören die Klassen automatisch zu Standard-Paket ohne best. Namen.

- Erst package ... ;

dann import ... ;
import ... ;
:

- Compilierung:

Haupt-Directory: Test.java
Enthält Unter-Directory listen.

Entweder:

javac listen/Liste.java

Oder

javac Test.java

(compiliert alle benötigten Klassen mit)

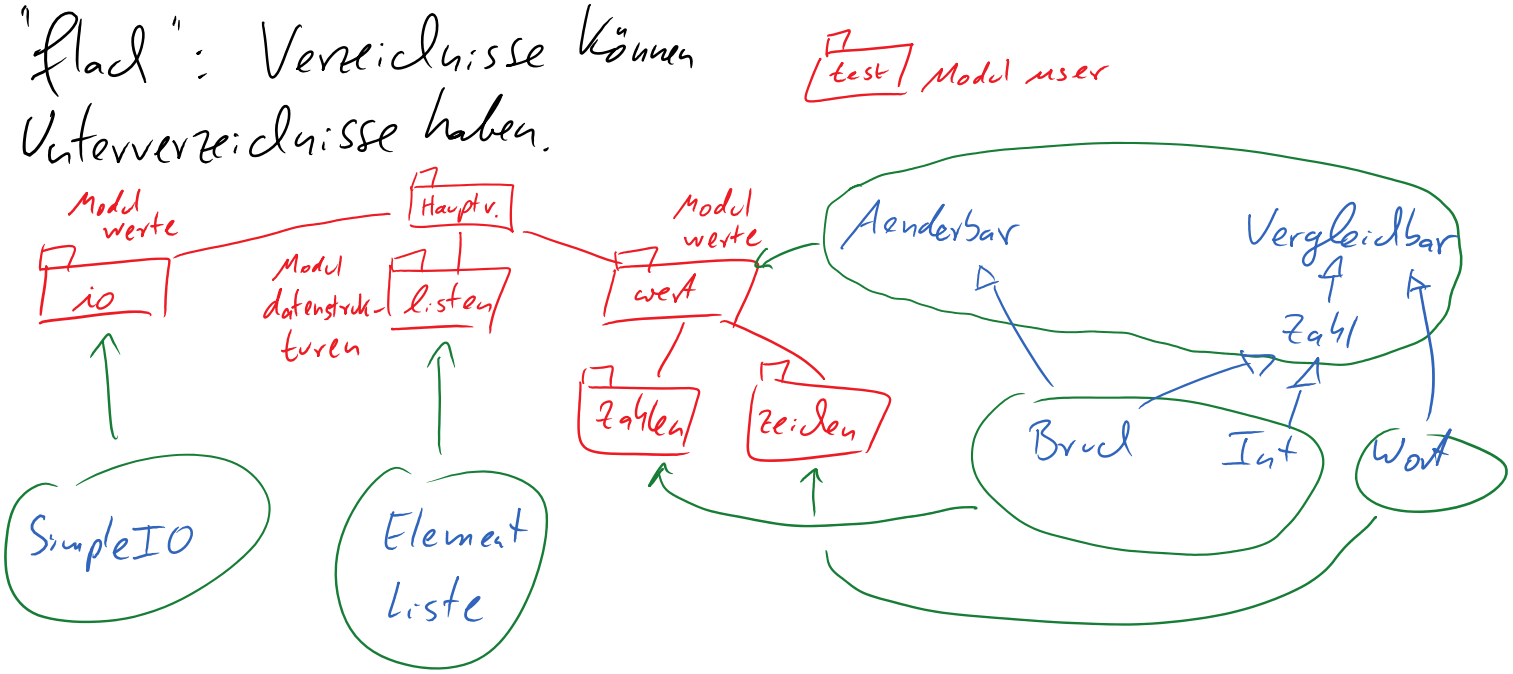
- Klassen "expatiren" alle

Funktionalität, die mit public, protected oder "kein Schlüsselwort" gekennzeichnet sind.

Pakete "exportieren" alle Funktionalität mit public oder protected.

⇒ Schnittstellendokumentation mit javadoc.

- Paket java.lang wird automatisch immer importiert (enthält Standardklassen wie Object).
- Paket-Struktur ist nicht "flach": Verzeichnisse können Unterverzeichnisse haben.



Wir haben jetzt 5 Pakete: io, listen, wert, wert.zahlen, wert.zeichen

- Logisch sind Unter- und Oberpakete unabhängig.

Dh: Man kann nicht automatisch auf Komponenten des Unter- oder Ober-Pakets zugreifen, sondern muss diese importieren.

- Pakete fassen Klassen + Interfaces zusammen.

Seit Java 9:
Module fassen Pakete zusammen.