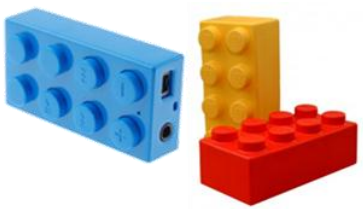
Not for public release!
copyright J. Rau, 2011

2. Modularisierung

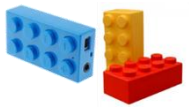
Modularisierung ist die Zerlegung eines Systems in einzelne relativ unabhängige Einzelteile (Module), die über genau definierte Schnittstellen miteinander verbunden werden.

Eine Klasse ist also ein besonders gutes Mittel zur Modularisierung von Softwaresystemen.



Not for public release!
copyright J. Rau, 2011

2. Modularisierung



Wozu dient Modularisierung von Software ?

Sie unterstützt folgende Eigenschaften eines Systems:

Verständlichkeit

Jede Entwurfseinheit muss unabhängig von den anderen verständlich sein.

Kombinierbarkeit

Entwurfseinheiten lassen sich durch Rekombination zu neuen Systemen zusammenfügen.

Lokalität

Eine Änderung des Entwurfs darf möglichst wenige Änderungen in anderen Entwurfseinheiten verursachen.

Parallele Entwicklung

Teile eines Systems können gleichzeitig von verschiedenen Entwicklern erstellt werden.

Not for public release!
copyright J. Rau, 2011

2. Modularisierung

<u>Eine</u> Definition des Modulbegriffs:

- ist die Zusammenfassung von Methoden und Daten zur Realisierung einer in sich abgeschlossenen Aufgabe.
- Die Kommunikation eines Moduls mit der Außenwelt darf nur über eine eindeutig spezifizierte Schnittstelle erfolgen.
- Zur Integration eines Moduls in das System darf keine Kenntnis der Implementierung erforderlich sein.
- Die Korrektheit eines Moduls muss ohne Kenntnis seiner Implementierung nachprüfbar sein.

Not for public release!
copyright J. Rau, 2011

2. Modularisierung

Umsetzung der Modularisierung durch das Geheimnisprinzip
(Information Hiding)

Allgemein:
Trennung von Schnittstelle u. Implementierung

Datenkapselung ist die am meisten verbreitete und wichtigste Form von Information Hiding. Die Datenstruktur wird verborgen.
Das wird durch das Setzen von Zugriffsattributen für Daten und Methoden realisiert. Man greift nur über eine Schnittstelle auf diese zu.

Schnittstelle: besteht aus Operationen und Daten, die den Umgang mit der Datenstruktur beschreiben.

Not for public release!
copyright J. Rau, 2011

2. Modularisierung

Rechte durch **Zugriffs-Attribute** (access modifier) festlegen.

Sichtbarkeit	innerhalb d. Package	abgeleitete Klassen	außerhalb d. Package
public (+)	sichtbar		
private (-)	unsichtbar		
default (~)	sichtbar	unsichtbar	
protected (#)	sichtbar (sind Freunde ☺)		unsichtbar

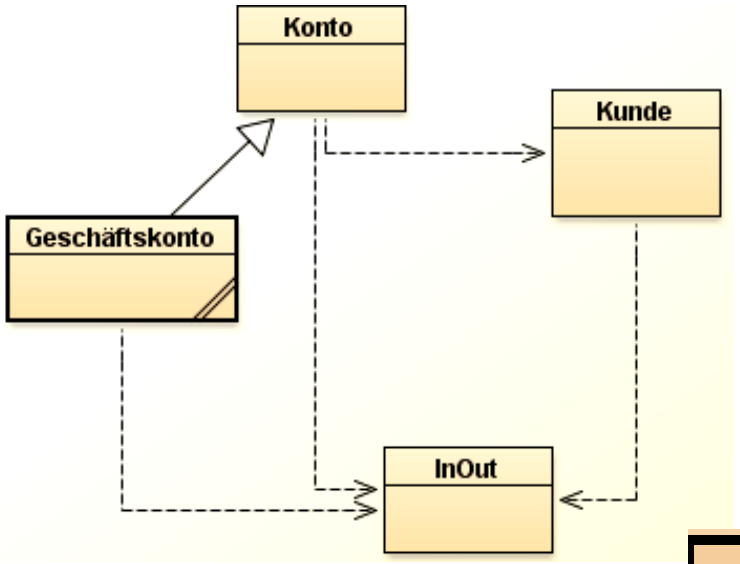
Packages:
Funktional zusammengehörende Klassen mit einheitlicher Schnittstelle.

Packages werden mit der Java-Plattform ausgeliefert (Standard-Packages) oder können selbst programmiert werden.

Not for public release!
copyright J. Rau, 2011

2. Modularisierung

Zugriffs-Attribute im UML



Kunde

~vorname: String
~nachname: String
~kontonummer: String
~wohnort: String

+ Kunde()

Konto

~kontonr: String
~inhaber: String
~ort: String
versuche: int
kontostand: double
pin: int
dispo: double
pintest: boolean

+ Konto()
+ getKontostand(): double
+ getInhaber(): String
+ kontoDatenEingeben(): void
# pruefePin(): boolean
# sperreKonto(): void
+ einzahlen(int): void
+ abheben(int): void
+ ueberweisen(Konto, int): void
+ druckeAuszug(): void