2 Datenabstraktion

2.1 Begriffe

Geheimnisprinzip:

(information hiding principle, Parnas 1972)

Zugriffe auf Teile einer Programmeinheit, die für die „reguläre Benutzung" nicht erforderlich sind, sollten **verboten** sein.

Besser:
Die nicht benötigten Teile sollten **unsichtbar** sein.

Datenabstraktion

Datenabstraktion: (data abstraction)

Anwendung des Geheimnisprinzips auf die Darstellung von (einfachen und komplexen) Daten im Rechner:

die Daten sind nicht durch Zugriff auf ihre Bestandteile, sondern **ausschließlich über zugehörige Operationen** manipulierbar.

[Auch „**Datenkapselung**" (data encapsulation)]

Warum Datenabstraktion?

Vorteile

1 Sicherheit:
Objekt kann nicht in ungültige Zustände gebracht werden.

2 Komfort:
Benutzer kann von Repräsentation abstrahieren;
Objekt wird ausschließlich über Operationen benutzt.

3 Flexibilität:
Code der Klasse kann unabhängig vom benutzenden Code entwickelt werden - und ohne Auswirkungen auf den benutzenden Code geändert werden.

Datenabstraktion und Objektorientierte Programmierung

2.2 Datenabstraktion in Programmiersprachen

Java

```
class Queue {
//Overview: .....
      private int head, length;
// Methods
void append(Entry item)throws QueueOverflow {
//requires      . . . . . }
     ...
}
Queue q = new Queue();
```

`Queue` ist abstrakter Datentyp (abstract data type, ADT)

`q: Queue` verweist auf abstraktes Datenobjekt (ADO)

Haskell

- Datenabstraktion in Haskell

```
module Queues(Queue, emptyq, append, ...)
where
data Queue t = Qrep[t]   -- Qrep NOT exported
 ...


module QueueTest where
import Queues
 ...
 ... append x q ...
```


Modulbasiert: Modula-2

```
DEFINITION MODULE Queues;          (* Modula *)
TYPE Queue;
PROCEDURE append(x: String; q: Queue);
PROCEDURE remove(q: Queue): String;
 ...
END Queues.
```

```
IMPLEMENTATION MODULE Queues;
TYPE Queue = POINTER TO Qrep;
     Qrep  = RECORD .... END;(* data
  representation *)

PROCEDURE append(x: String; q: Queue);
          ....................  (* procedure
  code *)
 ...
END Queues.
```

Java Packages

```
// Datei Queue.java
package queues;
 .....          // weiterer Java-Code


// Datei QueueImpl.java
package queues;
 .....          // weiterer Java-Code


// Datei Test.java
package queueTest;
import queues.*;
 .....          // weiterer Java-Code
```