

Benchmarking verschiedener Listenimplementationen

Ein Beispiel für ein ADT ist die [Verkettete Liste](#). Eine Liste enthält beliebig viele Werte eines Typs T, z.B. Integer-Zahlen. Zudem sind Operationen definiert, die mit der Liste durchgeführt werden können, z.B. `anhaengen(val: T)`. In der objektorientierten Programmierung entspricht das den Methoden.

Die Namen der Operationen und ihre Signatur (also Parameter und Rückgabewert) können in Java durch eine **abstrakte Oberklasse** oder ein **Interface** festgelegt werden – dies ist die **syntaktische** Beschreibung.

Allerdings sagen die Methoden und ihre Signaturen nichts über das **Verhalten** des ADTs aus, das ist die **Semantik**. Wir müssen also beschreiben, was die Methode `anhaengen(val: T)` tut. Prinzipiell könnte die Methode `anhaengen` einfach einen leeren Rumpf haben – syntaktisch wäre sie damit korrekt, da es sich um eine Methode ohne Rückgabewert handelt.

Ein Programmierer, der die Klasse verwendet, stellt sich aber etwas anderes unter "anhängen" vor. Üblicherweise bedeutet "anhängen", dass das neue Element hinter den vorhandenen Werten angefügt wird. Man könnte sich aber auch vorstellen, dass das neue Element vorne angefügt wird oder nur dann, wenn es noch nicht vorhanden ist oder an der richtigen Stelle im Bezug auf eine Sortierung...



(A1)

Beschreibe das Verhalten der folgenden Methoden im Kontext einer verketteten Liste so eindeutig wie möglich. Was genau sollen diese Methoden machen, was geben Sie zurück, welche Voraussetzungen müssen evtl. erfüllt sein, damit man sie auf die verkettete Liste anwenden kann?

1. `vorneAnfuegen(val: T)`
2. `erstesElement(): T`
3. `letztesElement(): T`
4. `findeErstesVorkommen(val: T): int`

From:
<https://info-bw.de/> -

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:adt:verkettete_liste:benchmarking:start?rev=1625132769

Last update: **01.07.2021 09:46**

