Informatik II (D-ITET)
Frühjahrssemester 2016
Prof. Friedemann Mattern
Dr. Wilhelm Kleiminger
Hossein Shafagh (shafagh@inf.ethz.ch)

ETH Zürich
Institut für Pervasive Computing
Gruppe für Verteilte Systeme
http://vs.inf.ethz.ch

Übungsserie Nr. 5

Ausgabe: 6. April 2016
Abgabe: 13. April 2016

Hinweise

Für diese Serie benötigen Sie das Archiv
`http://vs.inf.ethz.ch/edu/FS2016/I2/downloads/u5.zip`.
Die Klasse *list.List* implementiert eine verkettete Liste von *int*-Werten. Sie wird in allen Aufgaben dieser Serie verwendet.

1. Aufgabe: (7 Punkte) Einfach verkettete Listen

Die Klasse *u5a1.Lists* bietet eine Reihe von Methoden zur Verwendung von Listen an. Ihre Aufgabe ist es, diese Methoden zu implementieren. Mehr Informationen zu den Methoden finden Sie in der Dokumentation. *Wichtig*: Ihre Implementierung muss rekursiv sein. Das bedeutet insbesondere, dass Sie keine *while*- oder *for*-Schleifen verwenden dürfen.

- *add* (1 Punkt)
- *size* (1 Punkt)
- *sum* (1 Punkt)
- *last* (1 Punkt)
- *sublist* (1 Punkt)
- *valueAt* (1 Punkt)
- *index* (1 Punkt)

Hinweis: Die Methode *u5a1.Lists.toString* ist bereits implementiert und dient als Beispiel für eine rekursive Methode für Listen.

2. Aufgabe: (8 Punkte) Modifizierung von Listen

Die Klasse *u5a2.MutableLists* bietet eine Menge von Methoden an, um Listen zu verändern. Ihre Aufgabe ist es, diese Methoden anhand der vorhandenen Dokumentation (rekursiv) zu implementieren.

Hinweis: Verwenden Sie wenn möglich die von Ihnen implementierten Methoden aus *u5a1.Lists*.

- *append* (1 Punkt)
- *concat* (1 Punkt)
- *insertAt* (je 2 Punkte)
- *remove* (2 Punkte)

3. Aufgabe: (5 Punkte) Sortieren von Listen

Die Klasse *u5a3.SortedLists* bietet die Möglichkeit zum Sortieren von Listen. Ihre Aufgabe ist es, die Methoden anhand der vorhandenen Dokumentation rekursiv zu implementieren.

(3a) *(3 Punkte)* Implementieren Sie die Methode *insertSorted*, die einen Wert in eine bereits sortierte Liste so einfügt, dass die resultierende Liste ebenfalls sortiert ist.

(3b) *(2 Punkte)* Implementieren Sie nun die Methode *sort*, die unter Verwendung von *insertSorted* eine Liste sortiert.

4. Aufgabe: (6 Punkte) Noch ein wachsender Stack

Ein Nachteil des wachsenden Stacks aus Serie 4 ist, dass die Werte des Stacks immer wieder kopiert werden müssen, sobald die Kapazitätsgrenze des Stacks erreicht wird. Eine Alternative dazu ist es, einen wachsenden Stack mit Hilfe von einfach verketteten Listen zu realisieren. So ein Stack kann dynamisch wachsen, ohne jemals kopieren zu müssen.

Die Klasse *u5a4.Stack* soll diese Idee umsetzen. Ihre Aufgabe ist es, ihre Methoden anhand der vorhandenen Dokumentation zu implementieren.

- *push* (1 Punkt)
- *pop* (2 Punkte)
- *peek* (1 Punkt)
- *empty* (1 Punkt)
- *size* (1 Punkt)