2. Oktober 2014

Team: 02, Daniel Hofmeister, Alex Mantel

Aufgabenaufteilung:

1. Daniel Hofmeister: Skizze,Stack,Queue
2. Alex Mantel: Skizze,Liste,Array

Quellenangaben:

- Praktikumsaufgabe_1

Bearbeitungszeitraum:

- Skizze gemeinsam: 1h
- Alex Mantel: 4h
- Daniel Hofmeister: 4h

Aktueller Stand: Komplett fertig & abgenommen.

Fragen an: **alex.mantel@haw-hamburg.de**
daniel.hofmeister@haw-hamburg.de

Änderungen/Klarstellungen im Bezug auf die implentierte Fremdskizze:

– Fehlerbehandlung: Bei der Zurückgabe von Elementen die out of bounds sind wird „nil“zurückgegeben.

2. Oktober 2014

Für alle ADTs gilt:

- Fehler werden Ignoriert und die Eingabe wird zurück gegeben
- Die ADTs sollen als 2-Tupel dargestellt werden. An erster Stelle befindet sich als Atom der Typ {stack, ADT_LISTE}, {list, TUPEL}, {queue, {stack, stack}}, {array, ADT_LISTE}
  So lässt das Patternmatching von Erlang zu, dass wir zwischen den ADTs unterscheiden.
- Grundsätzliche Fehlerbehandlung: Wenn ADT Mutationen nicht möglich sind, soll der eigegebene ADT zurück gegeben werden.

Liste (List)

Zu implementierende Funktionen:

create: $\varnothing \rightarrow$ list
isEmpty: list $\rightarrow$ bool
laenge: list $\rightarrow$ int
insert: list × pos × elem $\rightarrow$ list
delete: list × pos $\rightarrow$ list
find: list × elem $\rightarrow$ pos[5]
retrieve: list × pos $\rightarrow$ elem[4]
concat: list × list $\rightarrow$ list

1. Die Datei bzw. das Modul soll *list.erl* heißen
2. Position des ersten Elementes ist 1
3. Eine leere Liste hat 0 Elemente.
4. Die Liste ist nicht destruktiv, falls ein Element an die Position eines bereits vorhandenen Elementes hinzugefügt wird, sollen alle folgenden Elemente um eine Position verschoben werden.
5. Bei Out of Range Zugriffen soll das atom 'nil' zurück geben werden
6. Die Liste arbeitet nicht destruktiv, d.h. wird ein Element an einer vorhandenen Position eingefügt, wird das dort stehende Element um eine Position verschoben.
   Retrieve soll die Liste nicht verändern.
7. Wenn das Element bei find nicht vorhanden ist, ist der Index -1
8. Die Implementation soll durch Tupel erfolgen. D.h. bei {list, X} soll X rein aus Tupel zusammen gesetst werden. Genauere Spezifikation wird dem Entwickler überlassen.

2. Oktober 2014

Stapel (Stack)

Zu implementierende Funktionen:

isEmpty:	list → bool
createS:	∅ → stack
push:	stack × elem → stack
pop:	stack → stack
top:	stack → elem
isEmptyS:	stack → bool

1. Datei soll *stack.erl* heißen
2. Der Stack ist ein LIFO (Last-in First-out)
3. Stack soll über die ADT Liste implementiert werden
4. Der Stack soll genau dann leer sein, wenn die interne Liste leer ist. D.h. hier soll das isEmpty deligiert werden.
5. Top soll den Stack nicht verändern.
6. Wenn der Stack leer ist soll das Atom 'nil' zurück gegeben werden

2. Oktober 2014

Schlange (Queue)

Zu implementierende Funktionen:

createQ:	∅ → queue
front:	queue → elem (Selektor)
enqueue:	queue × elem → queue
dequeue:	queue → queue (Mutator)
isEmptyQ:	queue → bool

1. Die Datei soll *queue.erl* heißen
2. Die Queue ist ein FIFO (First-in First-out)
3. Die Queue soll aus den ADT Stack erstellt werden
4. Es werden zwei Stacks verwendet: der In-Stack und der Out-stack
5. Bei Bedarf werden alle Elemente aus dem In-Stack in den Out-Stack geworfen. Bedarf ist denau dann, wenn der Out-Stack leer ist und auf diesen zugegriffen wird. Also bei front und dequeue
6. Die Queue ist genau dann leer, wenn beide Stacks leer sind
7. Front soll die Liste nicht verändern

2. Oktober 2014

Feld (Array)

Zu implementierende Funktionen:

initA:	$\varnothing \rightarrow$ array
setA:	array × pos × elem → array
getA:	array × pos → elem
lengthA:	array → pos

1. Die Datei soll *array.erl* heißen
2. Das erste Element soll auf dem Index 0 liegen.
3. Alle Plätze des Arrays ist mit 0 initialisiert
4. Ein Element wird überschrieben, falls bereits ein anderes Element auf dem gegebenen Index liegt.
5. Das Array hat keine feste Größe
6. Bei Out of Range Zugriffen wird eine 0 zurückgegeben.
7. Die Größe des Arrays ist der größte Index eines eingefügten Elementes.
8. GetA soll das Array nicht verändern