Aufgabe 1: Die Kunst der Abstraktion

In dieser Aufgabe werden wir unterschiedliche ADT's gemäß einer vorgegebenen Skizze implementieren. Die Skizze wird aus einer innerhalb der Praktikumsteilnehmer_innen erstellten Skizze ausgewählt. Ziel ist, die Kunst der Abstraktion zu üben. Daher sind die Programme so zu gestalten, dass die jeweiligen ADT's einzeln mit anderen Teams getauscht werden können, z.B. tauscht Team 06 mit Team 03 die ADT Stack, ohne jedoch die ADT Liste zu tauschen.

Aufgabenstellung

Die exakten technischen Vorgaben (syntaktische Signatur, Dateinamen etc.) werden in einer Skizze von einem Team beschrieben werden. Zur Fehlerbehandlung: Sollten
nicht vorhandene Elemente gelöscht werden, in der Liste an unmöglicher Stelle eingefügt werden, etc. ist die Fehlerbehandlung durch „Ignorieren“ durchzuführen, d.h. es wird so gehandelt, als hätte die Operation nicht stattgefunden. Aus Dokumentationsgründen können log-Dateien erstellt werden.

Eine **ADT Liste** ist zu implementieren:

Vorgabe:
Funktional (nach außen)

1. Die Liste beginnt bei Position 1;
2. Die Liste arbeitet nicht destruktiv, d.h. wird ein Element an einer vorhandenen Position eingefügt, wird das dort stehende Element um eine Position verschoben.
3. Die Elemente sind vom Typ „ganze Zahl“.

Technisch (nach innen)

1. Der ADT Liste ist intern mittels eines Java-Arrays zu realisieren (…new int[?]…)

Objektmengen: pos, elem, list
Operationen: (semantische Signatur)

create: Ø ? list
isEmpty: list ? bool
laenge: list ? int
insert: list × pos × elem ? list
delete: list × pos ? list
find: list × elem ? pos
retrieve: list × pos ? elem
concat: list × list ? list

Eine **ADT Stack** ist zu implementieren:

Vorgabe:
Technisch (nach innen)

1. Der ADT Stack ist mittels ADT Liste zu realisieren.

Objektmengen: elem, stack
Operationen: (semantische Signatur)

createS: Ø ? stack
push: stack × elem ? stack
pop: stack ? stack
top: stack ? elem
isEmptyS: stack ? bool

Eine **ADT Queue** ist zu implementieren:

Vorgabe:
Technisch (nach innen)

1. Der ADT Queue ist mittels ADT Stack, wie in der Vorlesung voirgestellt, zu realisieren. Es sind z.B. zwei explizite Stacks zu verwenden und das „umstapeln“ ist nur bei Zugriff auf einen leeren „out-Stack“ durchzuführen.

Operationen: (semantische Signatur)

createQ: Ø ? queue
front : queue ? elem (Selektor)
enqueue : queue × elem ? queue
dequeue : queue ? queue (Mutator)
isEmptyQ : queue ? bool

Eine **ADT Array** ist zu implementieren:

Vorgabe:
Funktional (nach außen)

1. Das Array beginnt bei Position 0.
2. Das Array arbeitet destruktiv, d.h. wird ein Element an einer vorhandenen Position eingefügt, wird das dort stehende Element überschrieben.
3. Die Länge des Arrays wird bestimmt durch die bis zur aktuellen Abfrage größten vorhandenen und explizit mittels beschriebenen Position im array.
4. Das Array ist mit 0 initialisiert, d.h. greift man auf eine bisher noch nicht beschriebene Position im Array zu erhält man 0 als Wert.
5. Das Array hat keine Größenbeschränkung, d.h. bei der Initialisierung wird keine Größe vorgegeben.

Technisch (nach innen)

1. Der ADT Array ist mittels ADT Liste zu realisieren.

Objektmengen: pos, elem, array
Operationen: (semantische Signatur)

initA: Ø ? array
setA: array × pos × elem ? array
getA: array × pos ? elem
lengthA: array ? pos

Zudem sind für die jeweilige ADT **JUnit-Tests** zu implementieren, die einfach ausgetauscht werden können (als *.jar z.B. adtlisteJUt.jar speichern, siehe Vorgabe in der Skizze). Beachten Sie: die Tests sollten auch Belastungen (große Datenmengen) und Grenzfälle beinhalten. Die Tests werden ausgetauscht werden.

Abnahme

Da die Aufgaben des Praktikums sehr frühzeitig im WWW zur Verfügung stehen, wird die Abnahme stark auf eine **vorbereitende Arbeit** aufgebaut.

Bis Sonntag Abend 20:00 Uhr zwei Wochen vor Ihrem Praktikumstermin ist eine erste Skizze der Aufgabe als *.pdf Dokument (Dokumentationskopf nicht vergessen!) mir per E-Mail zuzusenden (Gruppe 1 11.10; Gruppe 2 18.10; Gruppe 3 25.10). Ggf. können offene Fragen mit gesendet werden. Die Skizze **muss** grob beschreiben, wie Sie sich die Realisierung denken. Als erfolgreich wird eine Skizze bewertet,

wenn Ihre Kenntnisse bzgl. der gestellten Aufgabe eine erfolgreiche Teilnahme an dem Praktikumstermin in Aussicht stellen.

Am Tag des Praktikums findet eine Befragung von Teams statt. Die **Befragung muss erfolgreich absolviert werden**, um weiter am Praktikum teilnehmen zu können. Ist die Befragung nicht erfolgreich, gilt die Aufgabe als nicht erfolgreich bearbeitet. Als erfolgreich wird die Befragung bewertet, wenn Ihre Kenntnisse bzgl. der gestellten Aufgabeausreichend oder besser sind. Dazu gehört insbesondere eine mindetsens ausreichende Kenntnis über Ihren gesamten Code. Bei der Befragung handelt es sich nicht um die Abnahme.

Abgabe: Unmittelbar am Ende des Praktikums, spätestens bis 19:00 Uhr am selben Tag, ist von allen Teams der Code abzugeben. Zu dem Code gehören die Sourcedateien, die ggf. erzeugten *.log etc. Dateien, die während der Tests erzeugt wurden, und eine Readme.txt Datei, in der ausführlich beschrieben wird, wie die Software zu starten ist! Zudem ist der aktuelle Dokumentationskopf abzugeben. Die Dateien sind als *.zip Ordner (mit cc an den/die Teampratner_in) per E-Mail abzugeben. Die Abgabe gehört zu den PVL-Bedingungen und ist einzuhalten, terminlich wie auch inhaltlich!

Wird eine Aufgabe nicht erfolgreich bearbeitet, gilt die **PVL** als **nicht bestanden**. Damit eine Aufgabe als erfolgreich gewertet wird, muß die Befragung als erfolgreich gewertet werden sowie die Abgabe abgenommen worden sein. **Alle gesetzten Termine sind einzuhalten**. Dies ist notwendig, da sonst erhebliche zeitliche Verzögerungen stattfinden würden..