Übungsblatt 2

Ausgabe: 10.04.2014
Abgabe: 07.05.2014

Aufgabe 1: Interfaces — 40 Punkte

Bauen Sie die `Queue` und den `Stack`, die sie in ADS kennengelernt haben zu ADTs um (Die Funktionalität von Stack und Queue wird erweitert um eine Methode `size`, welche die Anzahl der Elemente in `Stack` bzw. `Queue` zurückgibt) und organisieren sie Ihre ADTs in einem Paket namens `myutil`.

Verstecken sie private Daten in den Klassen, berücksichtigen sie aber dabei, dass Unterklassen auf diese Daten zugreifen können sollen. Dabei sind `Stack` und `Queue` als `Interfaces` zu definieren. Überlegen Sie genau **welchen Rückgabetyp** die Methoden von `Stack` und `Queue` haben (müssen), damit Sie auf besondere Situationen reagieren können.

Realisieren sie **jeweils zwei** Implementierungen von `Stack` und `Queue`:

a) Mittels des ADT `LinkedList` (aus Übung 1). Auch, wenn eine Liste beliebig viele Elemente aufnehmen kann, sollten sie die Anzahl der Elemente beschränken.
b) Implementieren sie `Stack` und `Queue` mittels eines wachsenden Arrays, d.h. sobald `Stack` bzw. `Queue` überlaufen, wird das zugrunde liegende Array verdoppelt. Das Verdoppeln soll nur einmalig vorgenommen werden. Läuft das Array erneut über, kann nichts mehr eingetragen werden. `Stack` und `Queue` sollen 2 Konstruktoren haben. Einen parameterlosen, der einen Puffer mit Standardgröße (die kann frei gewählt werden) anlegt und einen Konstruktor, welchem die Anzahl der Elemente übergeben wird.

Hinweise:
Überlegen sie, welche Maßnahmen bei der Verdoppelung des Arrays notwendig sind. Die Queue wird als **Ringpuffer** organisiert. Bei der Verdopplung des Arrays muss auf die korrekte Reorganisation des Ringpuffers geachtet werden.
Die Elemente, die in `Stack` und `Queue` eingetragen werden sind vom Typ `Object`. Im Testprogramm werden dann `int`- Werte oder Strings eingetragen. Sie dürfen bei der Listenimplementierung von `Stack` und `Queue` davon ausgehen, dass nur `IntListNodes` und `StringListNodes` (wie in Übungsblatt1 definiert) eingetragen werden.

Schreiben sie ein Testprogramm, welches `myutil.Stack` und `myutil.Queue` verwendet. Die Tester sollen nach Lust und Laune Werte in `Stack` und `Queue` ablegen bzw. herausholen können.

Schreiben sie ihr Testprogramm so, dass Sie zwischen den beiden Implementierungen der Schnittstellen wechseln können.
Schreiben Sie zudem sinnvolle JUnit-Tests.

Aufgabe 2: Interfaces für Callback-Simulation (am Beispiel von Sortierverfahren) 40 Punkte

Nehmen Sie die beiden internen Sortierverfahren ShakerSort und InsertionSort aus ADS (wenn sie andere Sortierverfahren korrekt implementiert haben, dann dürfen Sie auch diese nehmen).

a) Ändern sie Sortierverfahren dahingehend, dass Sie nun nicht nur `int`-Arrays sortieren, sondern Arrays vom Typ `Comparable`. Konkret werden in einem `Comparable`-Array dann `MyInt`-Objekte und `MyString`-Objekte (diese beiden Klassen sind noch zu definieren) abgelegt.

b) Schreiben Sie eine statische Methode `sortArray`, welche als Parameter ein Sortierverfahren und ein zu sortierendes Array mit `Comparable`-Werten bekommt. Schreiben Sie ein Hauptprogramm, in welchem Sie ihr Eingabefeld mit unterschiedlichen Verfahren sortieren lassen, indem sie das Eingabefeld und das gewünschte Verfahren als Parameter an `sortArray` übergeben. `sortArray` soll auch das Eingabefeld vor und nach dem jeweiligen Sortiervorgang ausgeben.

Zur Initialisierung der Felder mit den Objekten vom Typ `MyInt` bzw. `MyString` schreiben Sie am besten eine statische Methode, die ein mit `int`-Werten bzw. `Strings` direkt initialisiertes Array in ein entsprechendes Array vom Typ `MyInt` bzw. `MyString` überträgt.