Aufgabe 3.1 (Theorie)
Wie entwickelt sich im Speicher, in bildlicher Darstellung, ein dynamisches Array (Halbierung der Arraylänge bei Viertelfüllung, Verdopplung im Vergrößerungsfall, mit Startspeichergröße 2) unter folgenden Operationen?

`add(1); add(2); add(3); remove(); remove(); add(4); add(5); remove(); add(6); add(7); add(8); add(9); remove();`

Geben Sie jeweils sowohl size als auch length des Arrays und den Inhalt aller Array-Komponenten an und zeigen Sie jeweils deren aktuellen Zustand (lassen Sie keine Felder leer), beginnend mit dem Startzustand. Markieren Sie uninitialisierte Array-Elemente mit x. Wenn bei einer Aktion Feldelemente verschoben werden müssen, lassen Sie nicht mehr gebrauchte Feldelemente unverändert, solange diese nicht neu gebraucht werden.

Aufgabe 3.2
Ergänzen Sie die Klasse DynArray um folgende Methoden: (Die Methode `decrease()` sollte als **private** deklariert sein, alle anderen Methoden sind öffentlich. Sie dürfen gerne weitere *nicht-öffentliche* Hilfsmethoden definieren und nutzen.)

a) **void** remove(**int** pos)
löscht das Element an der Position pos aus dem Array, indem alle folgenden Elemente um eine Position nach vorne rutschen.
Bei Übergabe eines unzulässigen Werte für pos soll die Methode nichts tun.

b) **void** remove()
löscht das Element an der Position 0 aus dem Array (falls das Array nicht leer ist).

c) **void** delete (T e)
löscht das (das erste Vorkommen des) Element(s) $\in$ aus dem Array, sofern es enthalten ist. Wie bei a) sollen alle nachfolgenden Elemente um eine Position nach vorne rutschen.
Falls das Element gar nicht im Array enthalten ist, soll die Methode nichts tun.

d) **boolean** contains (T e)
prüft, ob das Element e im Array enthalten ist.

e) **void** decrease ()
reduziert die Array-Länge auf *die Hälfte* wenn das Array nur noch zu *einem Viertel* gefüllt ist (aber niemals auf weniger als die Startgröße, bei uns also: 2).
Beispiel: Array-Länge = 32 size = 8 (oder weniger): $\rightsquigarrow$ Reduktion der Array-Länge auf 16

f) Schreiben Sie JUnit-Tests, um Ihre Methoden auf korrektes Verhalten zu testen. Welche Fälle sollten geprüft werden? Als Testfälle zum Beispiel für `delete(T e)` sollten Sie (mindestens) berücksichtigen:

- e kommt irgendwo „in der Mitte des Arrays“ vor
- e kommt mehrmals vor
- e ist das erste Element des Arrays
- e ist das letzte Element des Arrays
- e kommt gar nicht vor

g) Diskutieren Sie: Warum wäre es keine gute Idee, zum Löschen eines Elementes die entsprechende Array-Komponente einfach auf **null** zu setzen?

Aufgabe 3.3
Implementieren Sie das Interface Menge<T> in einer Klasse MengeDynArray mithilfe eines dynamischen Arrays (statt wie in MengeLimited mit einem Array fester Größe).

Sie finden den Code des Interface Menge (und der Klasse MengeLimited) im LEA-Kurs im Ordner `U02Menge`.

Aufgabe 3.4

a) Implementieren Sie eine Klasse Modul:
ein Modul hat eine (String) Bezeichnung und eine (int) Anzahl von ECTS-Punkten.

b) Außerdem hat eine Modul ein Instanzattribut teilnehmer, das die Menge aller angemeldeten Studis darstellt.
($\rightsquigarrow$ Was ist eine sinnvolle Typvereinbarung für dieses Attribut?)

c) Modifizieren Sie die Klasse Stud aus Kap 1 (oder implementieren Sie sie neu):
Studis haben (wie bisher) einen Namen (String) und eine Matrikelnummer, die bei Erzeugung des Objektes fortlaufend ab 100 erzeugt wird.

d) Studis verfügen über eine (Instanz-)Methode **void** anmelden(Modul m), mit der sie sich zu einem Modul anmelden können. Ebenso soll es eine Methode **void** abmelden(Modul m) geben, mit der sich das Stud-Objekt auch von einem Modul wieder abmelden kann.

e) Außerdem soll es in Stud eine (Instanz-)Methode belegt () geben, die die Menge der Module liefert, für die das Stud-Objekt angemeldet ist.
($\rightsquigarrow$ Was ist ein sinnvoller Rückgabetyp dieser Methode?
Bedenken Sie außerdem:
Durch das An- bzw Abmelden eines Studis zu bzw von einem Modul ändert sich die Menge der belegten Module des Stud-Objektes **und** die TN-Menge des Moduls!)

f) Schreiben Sie eine Test-(Main)-Klasse, in der Sie verschiedene Studis und verschiedene Module erzeugen und einen Verlauf simulieren: Studis melden sich zu (mehreren) Modulen an oder wieder ab, mehrere Studis belegen das gleiche Modul usw.
Lassen Sie sich die Beleg-Listen der Studis und die Teilnehmerlisten der Module auf dem Monitor anzeigen.

Hinweis:
Wenn Sie A3.2. und A3.3 gut gelöst haben, bietet es sich an die Klasse MengeDynArray zu benutzen. Sie können aber auch mit MengeLimited arbeiten, wenn Sie bei den vorangehenden Aufgaben zu keinem guten Ergebnis gekommen sind.