ALGORITHMS & DATASTRUCTURES

LES 2 : BASIS DATASTRUCTUREN

THEORIE

Actie	**Hoofdstuk**	**2010**	**2012**
Globaal doornemen	The collections api	6.1 t/m 6.6	6.1 t/m 6.6
Bestudeer	Inner classes and implementation of ArrayList	15.5	14.5
Bestudeer	Stacks and queues	16	15
Bestudeer	Linked lists	17 (behalve 17.4)	16 (behalve 16.4)

THEORIE OPGAVEN

OPGAVE 1 (6.1AB)

Show the results of the following sequence: `add(2)`, `add(7)`, `add(0)`, `add(4)`, `remove()`, `remove()` when the `add()` and `remove()` operations correspond to the basic operations in the following:

a. Stack
b. Queue

OPGAVE 2 (16.1)

Given the following sequence: `add(2)`, `add(1)`, `remove()`, `add(4)`, `add(3)`, `remove()`, `remove()`, `add(5)`.

a. Show all steps of this sequence for a stack implemented as a linked list.
b. Show all steps of this sequence for a stack implemented as an array.
c. Show all steps of this sequence for a queue implemented as a linked list.
d. Show all steps of this sequence for a queue implemented as an array.

Assume an initial size of 4 for the array implementations.

OPGAVE 3 (17.1)

Draw an empty doubly linked list that uses both a header node and a tail node.

OPGAVE 4 (17.2)

Draw an empty linked list with header implementation.

PRAKTIJK

Op de ELO vind je de uitgangssituaties voor de verschillende datastructuren die je gaat implementeren. Houd je aan deze interfaces.

Er zijn unit testen opgeleverd om je datastructuren te testen. Over het algemeen schrijven we unit testen om de gewenste functionaliteit van een klasse te testen. Voor dit vak echter willen we soms ook controleren of de implementatie correct is uitgevoerd. Daarom zullen er ook unit testen zijn die *de inhoud* van je implementatie test. Daarom dienen voor sommige datastructuren de properties `public` te zijn gedefinieerd. Bedenk echter dat dit niet common practice is!

Probeer bij het implementeren van de datastructuren eerst zelf de klassen te maken voor je het boek erbij pakt!

OPDRACHT 1 ARRAYLIST

a) Implementeer een eenvoudige klasse `MyArrayList` waarin alleen getallen kunnen worden opgeslagen. Deze klasse is een implementatie van het interface `IMyArrayList`. Intern wordt een primitieve array bijgehouden. De list heeft een vaste capaciteit (grootte van de interne array) die wordt meegegeven aan de constructor. Deze capaciteit kan dus niet meer veranderen. Controleer op ongeldige bewerkingen en gooi zo nodig exceptions.
b) Implementeer ook een `ToString()`. Deze method retourneert `NIL` in het geval van een lege lijst. Bij een niet lege lijst retourneert de method de lijst met getallen, gescheiden door comma's, ingesloten door blokhaken (bijvoorbeeld `[3]` of `[1,2,3]`).
c) Test je datastructuur met behulp van de bijgeleverde unittests.
d) Geef de Big-Oh analyse van alle methods.

OPDRACHT 2 LINKED LIST

a) Implementeer een eenvoudige *singly* linked list `MyLinkedList` volgens het bijgeleverde interface. Het gaat hier om een *generic* klasse! De klasse heeft een zogenaamde *header node* (zie 17.1.1). Begin met het maken van een klasse `Node<T>`.
b) Implementeer een `ToString()`. De werking van deze method is identiek aan de `ToString()` van `MyArrayList`.
c) Test je datastructuur met behulp van de bijgeleverde unittests.
d) Geef de Big-Oh analyse van alle methods.

OPDRACHT 3 STACK

Maak een klasse `MyStack<T>` volgens het gegeven interface. Gebruik hierbij als interne datastructuur de klasse `MyLinkedList` uit de vorige opgave. Test je datastructuur.

OPDRACHT 4 TOEPASSING STACK

a) Maak een method `bool checkBrackets(string s)`. Gebruik `MyStack` van de vorige opgave om van een string te controleren of de haakjes kloppen. Bijvoorbeeld:

   ((( ) ( ))) is een geldige string
   ( )) is geen geldige string

b) Maak een method `bool checkBrackets2(string s)` die naast () ook [] controleert, bijvoorbeeld:
   [(([ ])) ( )] is een geldige string
   ([ )] is geen geldige string

OPDRACHT 5 QUEUE

Implementeer de op een array gebaseerde klasse `MyQueue<T>`. Gebruik voor de implementatie de C# klasse `List<T>`. Gebruik de bijgeleverde unittests om je datastructuur te controleren.