Programmeren in Java opdracht 2 datastructuren...

EG Software Engineering

18 november 2020

Lees eerst de hele opdracht alvorens te beginnen.

Inleiding

In diverse programma's is het nodig om met grote aantallen gegevens te kunnen werken. Wanneer we in programmeertalen zoals C, C++, Java, etc. met grote aantallen gegevens willen werken (bijvoorbeeld: getallen, Strings, o.i.d.) lijkt de oplossing eenvoudig: We gebruiken arrays. Met arrays kunnen we immers grote aantallen van hetzelfde datatype aanmaken.
Het gebruiken van arrays heeft echter enkele belangrijke nadelen:

- Arrays kunnen vol raken.
- Een programmeur zal bij het gebruiken van een array *van te voren* moeten weten hoe groot het array moet zijn. Het is echter niet altijd mogelijk van te voren te weten hoeveel gegevens je programma maximaal zal gaan verwerken.

Een alternatief voor het gebruiken van arrays zijn de zogeheten *dynamische datastructuren*. Dat zijn constructies die het mogelijk maken met grote aantallen gegevens om te gaan zonder dat je van te voren weet hoeveel gegevens je maximaal zult moeten verwerken.

In deze opdracht ga je drie van de meest voorkomende datastructuren programmeren:

- de linked list
- de queue
- de stack

Opdracht 1: de linked list

Je gaat in deze opdracht een linked list programmeren. Voor het uitvoeren van bewerkingen op deze lijst gebruiken we zes muppets.
Programmeer de volgende klassen:

- `App` Deze klasse bevat de `main` methode.
- `Muppet` Deze klasse stelt een figuur voor uit de welbekende Muppet show.
- `LinkedList` Deze klasse is de linked list datastructuur.

De klass App

In deze klasse bevindt zich de `main` methode. In deze methode instantieer je:

- Een `LinkedList` object.
- Zes `Muppet` objecten, te weten:
  - Animal
  - Beaker
  - Gonzo
  - Kermit
  - Miss Piggy
  - Swedish Chef

Elke Muppet heeft een leeftijd:

Animal	9
Beaker	4
Gonzo	21
Kermit	1
Miss Piggy	16
Swedish Chef	7

De klasse Muppet

Deze klasse stelt uiteraard een Muppet voor en heeft de volgende fields:

- `String naam`: De naam van de Muppet.
- `int leeftijd`: De leeftijd van de Muppet.
- `Muppet next`: Een pointer naar de volgende Muppet.

Daarnaast heeft de klasse enkele handige constructoren, waarmee we bij het instantiëren onze Muppet meteen een naam en/of leeftijd mee kunnen geven. Ook heeft de `Muppet` klasse een print methode, waarmee we de naam en de leeftijd van de Muppet op het scherm afdrukken:

- `void print()`

Voorzie ten slotte de `Muppet` klasse van de juiste `set`- en `get` methodes.

De klasse LinkedList

Deze klasse vormt de linked list zoals die tijdens de lessen is uitgelegd. Deze klasse bevat de volgende fields:

- `int size`: Het huidige aantal Muppets in de lijst.
- `Muppet start`: De eerste Muppet in de lijst.

De klasse bevat verder de volgende methodes:

- `void print()` drukt de hele lijst en de grootte op het scherm af. Als de lijst leeg is, wordt "`Lijst leeg.`" op het scherm afgedrukt.
- `void push(Muppet m, int p)` plaatst `Muppet m` op plek `p` in de lijst.
- `Muppet pop(int p)` verwijdert de muppet die op plek `p` in de lijst staat en geeft deze terug.
- `Muppet pop(String s)` verwijdert de muppet met naam `s` uit de lijst en geeft deze terug.
- `Muppet peek(String s)` geeft een pointer terug naar de muppet met naam `s` in de lijst of `null` als deze niet bestaat.
- `Muppet peek(int n)` geeft een pointer terug naar de muppet op positie `n` in de lijst of `null` als deze niet bestaat.

- `int size()` geeft het huidige aantal muppets dat in de lijst staat terug.

Maak alle methodes foolproof (wat betekent dat?) en voeg waar nodig `set`- en `get` methodes toe.

De Linked List testen in de main methode

Test de werking van je linked list door in de `main` methode het volgende uit te voeren:

- Maak in de `main` methode een `LinkedList` en de zes `Muppet` objecten aan.
- Roep de `pop` en daarna de `print` methode aan van de `LinkedList`.
- Push alle muppets op de lijst, zodanig dat ze in alfabetische volgorde in de lijst staan.
- Print de lijst.
- Haal in één regel code Swedish Chef uit de lijst en zet hem vooraan (wij houden van zijn baksels).
- print de lijst.

Werkwijze en tips

- Het schrijven van de `push` en `pop` methodes is in deze opdracht het moeilijkst. Test deze uitvoerig, opdat je zeker weet dat ze doen wat ze moeten doen.
- Teken bij het schrijven van diverse methodes op een stuk kladpapier wat er in het geheugen van de computer omgaat, zodat je stap voor stap kunt "volgen" wat er gebeurt.
- Het helpt bij het verhelpen van fouten in een algoritme nog wel eens om `println()` statements op cruciale plekken in een algoritme te plaatsen en de waarde van belangrijke variabelen op het scherm af te drukken. Deze, veel gebruikte, debugging techniek staat bekend als *scaffolding*. Zodra je zeker weet dat je algoritme naar behoren werkt, haal je de `println()` statements weer weg...

Extra: de stack

Schrijf de klasse Stack. Een stack is een datastructuur die veel lijkt op de LinkedList, maar met één belangrijk verschil: De stack werkt volgens het l.i.f.o. principe.

De klasse Stack

Methodes:

- void push(Muppet m) Plaatst een muppet op de stack.
- Muppet pop() Verwijdert een muppet van de stack.
- print() Print de inhoud van de stack.
- int size() geeft het huidige aantal muppets dat in de stack staat terug.

Werkwijze en tips

- Denk goed na, voordat je de stack gaat programmeren. Hint: Het programmeren van een stack is niet veel werk!

De stack testen

Test je stack in de main methode door het volgende uit te voeren:

- Instantiëer een Stack object.
- Roep van je stack de pop() en daarna de print() methode aan.
- Haal Kermit, Beaker en Swedish Chef uit de Linked List en push ze *in die volgorde* op de stack.
- Print zowel de linked list als de stack.

Extra: de Queue

Schrijf de klasse Queue. Een queue is een datastructuur die veel lijkt op de LinkedList, maar met één belangrijk verschil: De queue werkt volgens het f.i.f.o. principe.

De klasse Queue

Methodes:

- `void push(Muppet m)` Plaatst een muppet op de queue.
- `Muppet pop()` Verwijdert een muppet van de queue.
- `print()` Print de inhoud van de queue.
- `int size()` geeft het huidige aantal muppets dat in de queue staat terug.

Werkwijze en tips

- Denk goed na, voordat je de queue gaat programmeren. Hint: Het programmeren van een queue is niet veel werk!

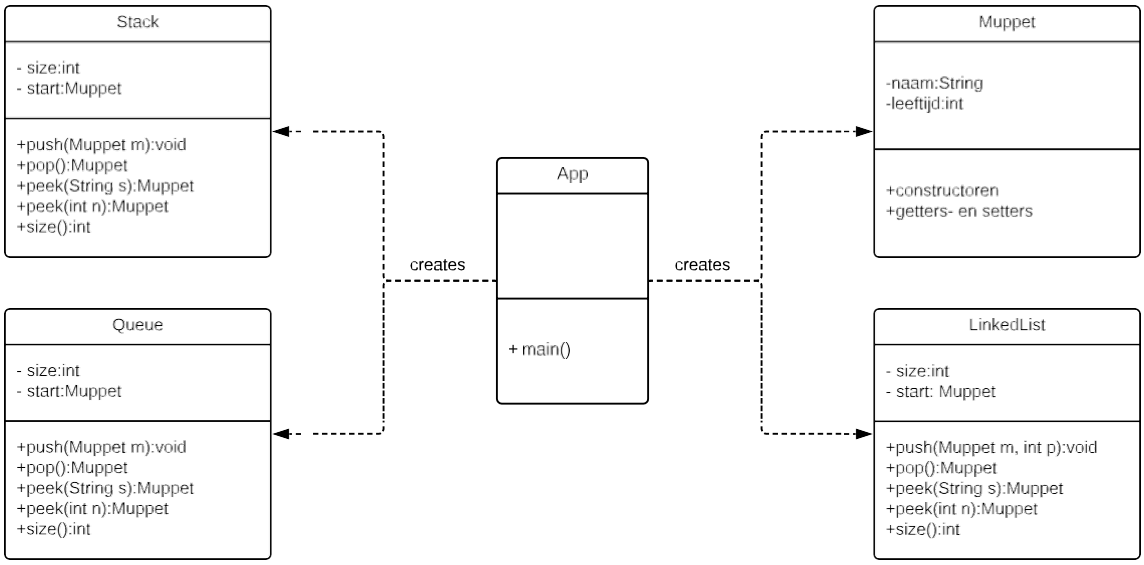
De queue testen

Test je queue in de `main` methode door het volgende uit te voeren:

- Instantiëer een `Queue` object.
- Roep van je queue de `pop()` en daarna de `print()` methode aan.
- Haal Gonzo, Miss Piggy en Animal uit de Linked List en push ze *in die volgorde* op de queue.
- Print zowel de linked list als de queue.
- Pop alle Muppets uit de queue en *daarna* alle Muppets uit de stack. Elke muppet die gepopt wordt, wordt *meteen* in de linked list gepushed op positie 1.
- Print je linked list.

Het UML diagram

In onderstaand UML diagram is weergegeven wat de diverse klassen met elkaar te maken hebben.



Eisen

- Voer in de `main` methode de bewerkingen uit zoals ze gegeven zijn in de Linked List, Stack en Queue opdrachten.
- Het gebruiken van ingebouwde datastructuren zoals `ArrayList`, etc. is verboden. Het gaat er in deze opdracht nu juist om dat je leert *zelf* datastructuren te schrijven en dat je enig gevoel ontwikkelt voor de onderliggende techniek.
- Het gebruiken van Exceptions is niet nodig en verboden.
- Geen van de methodes in de diverse datastructuren leidt bij aanroepen tot een `null pointer` exception. Als dat toch gebeurt, zit er blijkbaar een fout in je algoritme...

Beoordeling

Je wordt op het volgende beoordeeld:

- Correcte implementatie van de Linked List: 5.5
  - Overzichtelijke programmeerstijl
  - Zinvolle naamgeving van fields, variabelen, etc.
  - Correct gebruik van modifiers
  - Correcte implementatie van de voorgeschreven methodes
- Correct gebruik van commentaar: 0.5pt
- Efficiëntie van de algoritmen: 1pt
- Correcte implementatie van de Stack: 1.5pt
- Correcte implementatie van de Queue: 1.5pt