KU LEUVEN

Objectgerichte Softwareontwikkeling 2021-2022
Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Labsessie week 8
Arrays

Theorie
De theorie die je vóór deze labosessie moet verwerkt hebben, heb je gezien in week 3: collections 1.

Doel
In deze labosessie werken we met arrays, één van de datastructuren van Java om te werken met een collectie. We werken zowel met eendimensionale als tweedimensionale arrays.

Formaat
Elke labosessie is onderverdeeld in drie verschillende delen: startoefeningen, labo-oefeningen en thuisoefeningen. **Startoefeningen** zijn oefeningen waarvan we verwachten dat je ze thuis oplost, vóór de labosessie. Op die manier geraak je vertrouwd met de materie en check je of je de theorie van de hoorcolleges goed begrepen en verwerkt hebt. **Labo-oefeningen** zijn oefeningen die tijdens de labosessie worden behandeld. **Thuisoefeningen** zijn oefeningen die dieper ingaan op de materie. Ze zijn optioneel voor wie sneller werkt, of als extra oefeningenmateriaal bij het studeren/oefenen thuis. Neem contact op met jouw labo-assistent voor feedback op deze oefeningen.

Leerdoelstellingen
Na deze sessie moet je in staat zijn om de volgende concepten te begrijpen en te implementeren:

- Een array declareren en initialiseren
- Een array gebruiken: opvullen met elementen, door itereren met lussen etc.
- Meerdimensionale arrays kunnen gebruiken

KU LEUVEN

Objectgerichte Softwareontwikkeling 2021-2022

Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Startoefeningen

Open het project “StarterExercises” op Toledo bekijk de klasse Arrays.

1. Inspecteer de klasse en de constructor. De klasse heeft een array als instantievariabele en deze wordt opgevuld met de eerste 20 even getallen. Pas de constructor aan zodat het mogelijk is de lengte van de array te kiezen (via een parameter) bij het aanmaken van een Arrays-object. Vergeet niet ook de lus aan te passen!
2. Inspecteer de methodes *average()* en *findValue()* en zorg dat je de code begrijpt.
3. Vul de methode *printArray()* aan, zodat je de waarden één voor één kan printen. Je kan een *for*- of *for-each*-lus gebruiken.
4. Vul de methode *printArrayBackwards()* aan, zodat je de waarden één voor één kan printen maar nu startend vanaf het laatste element. Je print enkel achterwaarts, de array zelf wordt niet omgedraaid. Waarom kan je nu geen *for-each*-lus gebruiken?

KU LEUVEN

Objectgerichte Softwareontwikkeling 2021-2022

Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Labo-oefeningen

Oefening 1: Operaties met arrays

Open het "ArrayExercise"-project dat je kan downloaden van Toledo en vervolledig elke methode. Gebruik de testklasse om je oplossing te controleren.

Methode 1. Zoek de maximumwaarde van de elementen in de array en return deze.

Methode 2. Zoek en return de index van het *laatste* voorkomen van een gegeven element. Als het element niet voorkomt, return je -1. Als voorbeeld: [1,2,3,4,3], het te zoeken element: 3 → 4.

Methode 3. Vermenigvuldig alle elementen in een array met een gegeven integer en return de vermenigvuldigde array.

Methode 4. Voeg een element in op een gegeven index van de array. Als voorbeeld: index 2, getal 5 [1,2,3] →[1,2,5,3]. Dit lijkt te suggereren dat je de array op één of andere manier langer kan maken, maar dat lukt natuurlijk niet! Daarom definieer je voor deze methode best een *nieuwe* array als lokale parameter, en kopieer je de elementen van de parameterarray naar deze nieuwe (behalve natuurlijk op de plaats waar het nieuwe element ingevoegd moet worden). Je returnt dan de nieuwe array.

Methode 5. Keer de array om. Als voorbeeld: [1,2,3] →[3,2,1]. Ook hier is het beter om te starten van een nieuwe array, de elementen omgekeerd aan te vullen, en de nieuwe array te returnen.

Methode 6. Ga na of twee arrays gelijk zijn aan elkaar. Return een boolean.

Oefening 2: Multidimensionale Arrays

Arrays zijn niet alleen eendimensionale containerobjecten, maar kunnen zoveel dimensies bevatten als je wilt. Deze multidimensionale arrays kunnen worden gezien als arrays die arrays bevatten. Een tweedimensionale array kan je beschouwen als een matrix. Open het MultidimensionalArrays-project op Toledo en bekijk de demo()-methode. Voltooi de volgende methodes:

1. printArray() - Deze methode gaat over de hele array en drukt elke waarde af. Print "\n" om een nieuwe lijn in de console te starten.

2. hideAndSeek(). Deze methode vult willekeurig een array boolean[50][50] in met booleans. Zoek alle booleans true en sla het totale aantal op in de variabele answer. Je kan het seed-getal aanpassen om nieuwe arrays te genereren. De code die je feedback geeft staat mee in de methode, maar mag je negeren!

Oefening 3: Beschrijvende statistiek

Voor deze oefening is er geen startproject. Je startpunt is het bestand "results.txt", dat de examenresultaten van een vak bevat. Dit bestand bevat per lijn 1 resultaat dat bestaat uit een integer die tussen 0 en 20 ligt. Genereer een overzicht (hoeveel studenten per mogelijk resultaat) en toon deze info op het scherm. Om na te gaan of je code werkt kan je de opsomming van de resultaten in de file gebruiken:

Totaal aantal deelnemers = 343

Score : 0 behaald door 1 student

Score : 1 behaald door 0 studenten

Score : 2 behaald door 0 studenten

Score : 3 behaald door 0 studenten

Score : 4 behaald door 0 studenten

Score : 5 behaald door 2 studenten

Score : 6 behaald door 1 student

Score : 7 behaald door 6 studenten

Score : 8 behaald door 22 studenten

Score : 9 behaald door 25 studenten

Score : 10 behaald door 38 studenten

Score : 11 behaald door 35 studenten

Score : 12 behaald door 30 studenten

Score : 13 behaald door 40 studenten

Score : 14 behaald door 35 studenten

Score : 15 behaald door 34 studenten

Score : 16 behaald door 34 studenten

Score : 17 behaald door 23 studenten

Score : 18 behaald door 12 studenten

Score : 19 behaald door 4 studenten

Score : 20 behaald door 1 student

Tip1: als je 21 variabelen aan het definiëren bent en een hele lijst van if-/else-statements schrijft, ben je waarschijnlijk niet op de meest efficiënte manier bezig…

Tip2: kopieer gerust de code om een bestand uit te lezen uit de projecten van vorige week.

Oefening 4: Lotto

Start voor deze oefening van het "Lotto"-project dat je kan downloaden van Toledo. Gebruik de testklasse om je oplossing te controleren.

Vervolledig de klasse. De instantievariabele houdt 6 getallen bij. Vergeet ook de constructor niet aan te passen. Je implementeert verder twee methoden die getest worden: een getter voor de lottoNumbers en een void-methode die nieuwe lottonummers (1 tot en met 49) genereert. Tips:

- Bekijk de documentatie (online of de offline versie via de zip die je terugvindt op Toledo) van de Random-klasse.
- Je kan een willekeurig getal genereren en controleren of het al voorkomt in de array, indien wel hergenereer je het want er mag geen herhaling voorkomen.
- Je kan een aparte *private* methode schrijven om te controleren of het getal reeds voorkomt in de array.

KU LEUVEN

Objectgerichte Softwareontwikkeling 2021-2022

Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Thuisoefeningen

Oefening 1: Arrays Extra

Implementeer de volgende methodes. Tip: het merendeel van deze methodes vereisen het gebruik van geneste lussen.

Methode 1. Ga na of een gegeven array (doorgegeven als parameter) van karakters (char) gedupliceerde letters bevat. De methode returnt een boolean. Bonus: stop zodra je een gedupliceerde waarde hebt gevonden!

Methode 2. Zoek alle gemeenschappelijke elementen in twee arrays (dus twee parameters) van karakters (= doorsnede). Als voorbeeld: [‘a’, ‘b’, ‘c’] [‘a’,’b’,’d’]→[‘a’,’b’].

Methode 3. Schrijf een lus die een integer (doorgegeven als parameter) opsplitst in een array van de cijfers van die integer. Als voorbeeld: 15789 → [1,5,7,8,9].

Oefening 2: Weerstation bis

Maak een nieuwe versie van het weerstation van sessie 5-6. Hierbij houdt het weerstation een array bij van 5 thermometers. Schrijf methodes die je toelaten om waarden door te geven, informatie te printen enz.

Oefening 3: Game of Life[1] [uitdaging[2]]

Het “Game of Life” simuleert een bevolking van cellen in een tweedimensionale wereld. Ze worden geboren, leven, of sterven als gevolg van cellen rondom hen (elke cel heeft maximaal 8 buren). De regels zijn eenvoudig:

- Als een levende cel minder dan 2 levende buren heeft, sterft ze af uit eenzaamheid;
- Als een levende cel 2 of 3 levende buurcellen heeft, dan blijft ze leven;
- Als een levende cel meer dan 3 levende buurcellen heeft, dan sterft ze af door overbevolking;
- Als een dode cel juist drie levende buurcelen heeft, dan wordt ze levend (geboorte).

Om dit te simuleren, gebruikt het ‘Game of Life’ een matrix van cellen die levend of dood zijn. Voor een levende cel druk je een karakter af (bv. “#”), een dode cel blijft wit.

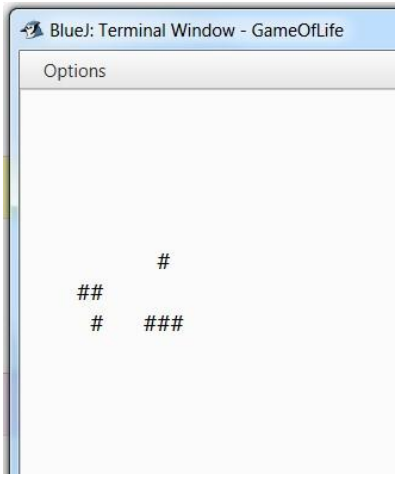
De startconfiguratie lees je in vanuit een configuratiebestand in volgend formaat:

> aantal rijen (r)
>
> aantal kolommen (k)
>
> r rijen met k kolommen met waarde 0 of 1, gescheiden door een spatie (1 is een levende cel, 0 een dode).

[1] De bedenker van de Game Of Life, de Engelse wiskundige John Conway, stierf vorig jaar aan de gevolgen van covid-19. Laat dit een eerbetoon wezen.
[2] Geen kleine…

Zoals in volgend voorbeeld (deze sequentie heeft 155 periodes alvorens ze uitsterft). Deze startconfiguratie kan je vinden op Toledo onder config1.txt.

```
12
20
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



Dit is de eerste generatie die je op het scherm afdrukt, uiteraard gebruik makend van een zelf geschreven methode. Vertrekkende van deze startsituatie worden bovenstaande regels gebruikt om het leven in de cellen van de volgende generatie te berekenen. Die matrix wordt op haar beurt getekend en dient weer als basis voor de volgende generatie, enz.

Schrijf je programma zo dat je een aantal gekozen generaties achtereenvolgens kunt tonen. Om ze als mens te kunnen beoordelen moet je natuurlijk je programma even kunnen pauzeren. Dit doe je door volgende lijnen code toe te voegen in je lus (het getal staat voor het aantal milliseconden dat je pauzeert, hier dus 1000 milliseconden of 1 seconde).

```
try
    {
       Thread.sleep(1000);
    }
catch(Exception e)
    {}
```

Nog handige code: System.out.println('\f') wist je consolevenster.