Les 2 Lesprogramma

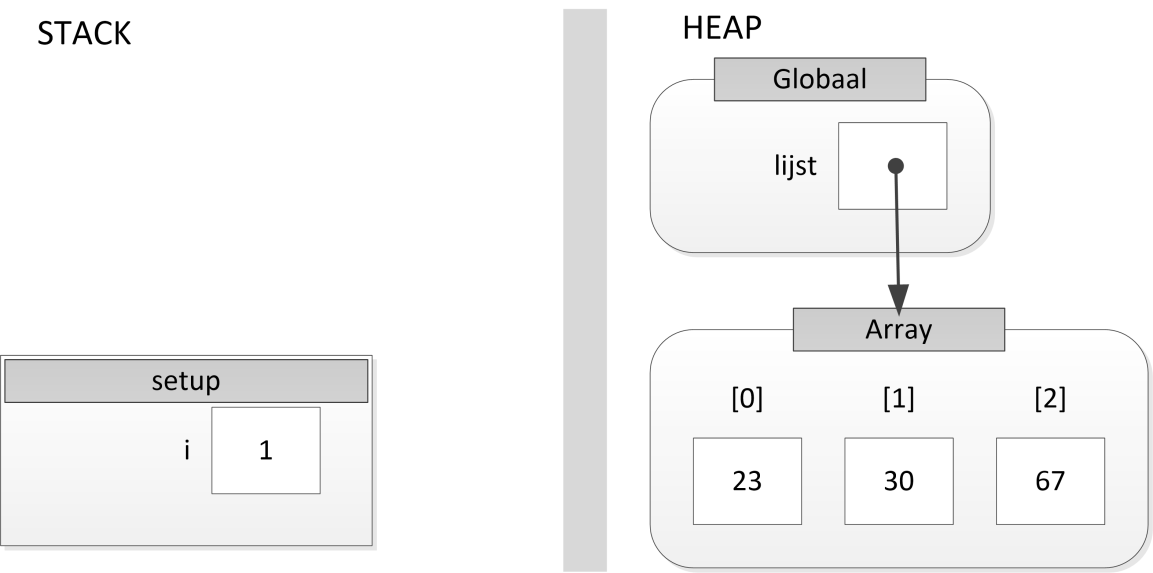
# Opgave Length

Klassikaal, begrip

Gegeven onderstaand programma

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07 | int[] lijst = {23, 1, 67};  void setup() {  for (int i = 0; i < lijst.length; i++) {  println(lijst[i]);  }  } |

Hieronder is het geheugenmodel te zien van het programma in de for-lus net na de aanroep van println op regel 5.



## Onderdeel A

Teken de variabele length met bijbehorende waarde op de juiste plek in dit model.

## Onderdeel B

Vergelijk de .-operator (in lijst.length) met de []-operator (lijst[i]). Wat is de overeenkomst tussen beide operatoren?

Tags: geheugenmodel, eigenschap, operatoren

# Opgave PlakCodeVoor

Individueel en klassikaal, programmeerstijl, programmeervaardigheid

Gegeven onderstaande stuk code.

String[] idLijst = {"8b3", "4bf", "9h0"};

void setup() {

println(plakCodeVoorId(idLijst[0], "NL"));

}

String plakCodeVoorId(String id, String code) {

return code + id;

}

In de variabele idLijst staan drie strings die gebruikt worden voor identificaties. In het hoofdprogramma wordt de code "NL" voor het eerste id geplakt uit idLijst. Hiervoor wordt de functie plakCodeVoorId gebruikt.

## Onderdeel A

Ga ervan uit dat er een methode bestaat plakCodeVoorIdLijst(String[] lijst, String code) die de string in de variabele code voor elk element uit lijst plakt.

Schrijf een test in setup waarmee je deze methode zou kunnen testen. Zie opgave “ Zoeken in Array” uit lesprogramma 1 voor de manier om een test op te zetten.

## Onderdeel B

Implementeer de methode plakCodeVoorIdLijst en zorg ervoor dat je de test uit opgave A slaagt.

## Onderdeel C

Direct klassikaal, of eerst in tweetallen

Vergelijk jouw implementatie met die van anderen. Welke oplossing is het meest stijlvol en welke oplossing het minst?

## Onderdeel D

Direct klassikaal, of eerst in tweetallen

Had de methode plakCodeVoorIdLijst(String[] lijst, String code) ook plakCodeVoorId(String[] lijst, String code) kunnen heten zonder een foutmelding te krijgen?

Tags: referentievariabele, array, overloading, new, return, geheugenmodel

# Opgave NotInitializedYet

Klassikaal, begrip

## Onderdeel A

Voer onderstaand programma uit en je krijgt een foutmelding.

void setup() {

int[] deLijst;

deLijst[0] = 1;

println(deLijst);

}

Wat is de foutmelding?

Is dit een runtimefout, of een compile-time foutmelding.

Teken het geheugenmodel op het moment dat regel 3 wordt uitgevoerd en leg aan de hand van dit model uit welke vervelende situatie deze foutmelding heeft voorkomen.

Tags: compile-time foutmelding <> runtime foutmelding, new, initialiseren, referentievariabele

# Opgave Product

Klassikaal, nieuwe stof

Bekijk onderstaande code:

void setup() {

String product1naam = "pc";

String product2naam = "mac";

int product1prijs = 500;

int product2prijs = 2000;

println(product1naam + " kost: " + product1prijs + " euro");

println(product2naam + " kost: " + product2prijs + " euro");

}

## Onderdeel A

We willen graag gebruik maken van een lus om alle producten te printen. Pas de code zo aan dat dit mogelijk is.

## Onderdeel B

Uitleg

Laten we naar één product kijken. We willen de eigenschappen naam en prijs groeperen. Dus we willen een object hebben met de eigenschappen naam en prijs:

void setup() {

??Type?? product1;

product1.naam = "pc";

product1.prijs = 500;

println(product1.naam + " kost: " + product1.prijs + " euro");

}

Vergelijk naam en prijs met de eigenschap length van array.

Het type van product1 moeten we zelf maken en dat doen we in een class (klasse). In deze klasse specificeren we ook alle eigenschappen:

## Onderdeel C

Maak een tweede product object voor de MAC.

Moeten we ook een nieuwe klasse maken?

Oefeningen

# Opgave Geheugenmodel van doeFunctie

In opgave doeFunctie, onderdeel A uit de vorige les heb je een runtime administratie van alle variabelen gemaakt tijdens de uitvoer van het programma. Schrijf deze administratie om naar een geheugenmodel.

Tags: stap voor stap doorlopen, runtime administratie, geheugenmodel

# Opgave Samenvatting geheugenmodel

De opgaven gaan over de onderstaande code:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | int[] deLijst;  int hetGetal;  void setup() {  hetGetal = 10;  deLijst = maakLijstMetEenGetal(2, hetGetal);  }  int[] maakLijstMetEenGetal(int lengte, int getal) {  int[] lijst = new Array[lengte];  for (int i = 0; i < lijst.length; i++) {  lijst[i] = getal;  }  return lijst;  } |

## Onderdeel A

Teken het geheugenmodel :

* na regel 2 en voor regel 4
* na regel 5 en voor regel 6
* tijdens de uitvoer van de methode op regel 6 en binnen deze methode na regel 9 en voor regel 10.
* tijdens de uitvoer van de methode op regel 6 en binnen deze methode na regel 13 en voor regel 14.
* na regel 6 en voor regel 7

## Onderdeel B

Hoe ziet de laatste versie van het geheugenmodel eruit als de declaratie van de variabelen deLijst en hetGetal in setup uitgevoerd wordt zoals hieronder te zien is:

|  |  |
| --- | --- |
| 01  02  03  04 | void setup() {  int hetGetal = 10;  int[] deLijst = maakLijstMetEenGetal(2, hetGetal);  }  // .. rest van de code weggelaten |

## Onderdeel C

Verklaar aan de hand van het geheugenmodel dat de methode setup niet bij de variabelen lengte, getal en lijst, maar de methode maakLijstMetEenGetal wel bij de variabele deLijst en hetGetal.

Gebruik in de uitleg zoveel mogelijk de technische begrippen die tot nu toe zijn behandeld.

## Onderdeel D

Is het, over het algemeen, verstandig om de methode maakLijstMetEenGetal gebruik te laten maken van deLijst en hetGetal.

Gebruik in de uitleg zoveel mogelijk de technische begrippen die tot nu toe zijn behandeld.