# Opgaves in de les

## Opgave length

### Onderdeel A

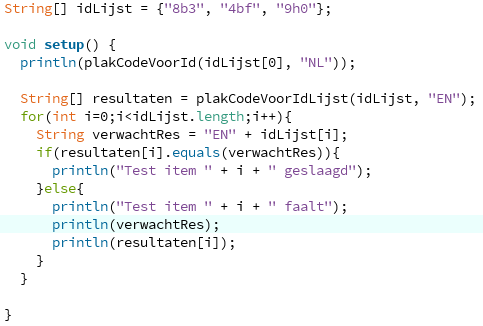
De length member staat in het array object die in de HEAP staat

### Onderdeel B

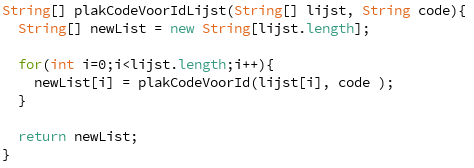
Het zijn beide eigenschappen van de array. Ook length is namelijk een referentie.

## Opgave PlakVoorCode

### Onderdeel A



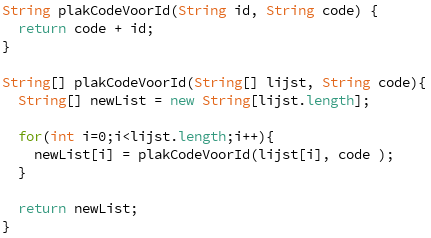
### Onderdeel B



### Onderdeel C

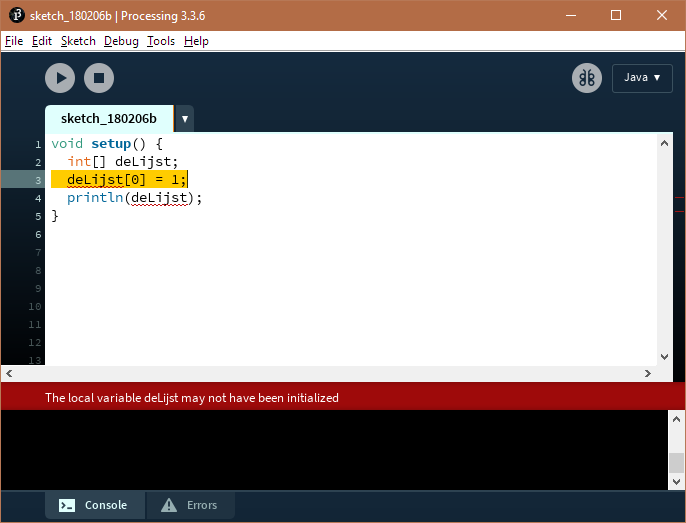
Maarten( een klasgenoot van mij ) heeft de zelfde uitwerking, alleen de variable naam voor de string is anders. Bij Maarten heet deze “updatedList”. Deze naam is eigenlijk wel beter omdat deze naam meer betrekking heeft op de variable.

### Onderdeel D

Ja dat kan, zoiets heeft een overload op een methode. De compiler gaat compiletime automatisch de juiste methode kiezen uit de beschikbare overloads. 

## Opgave NotInitializedYet

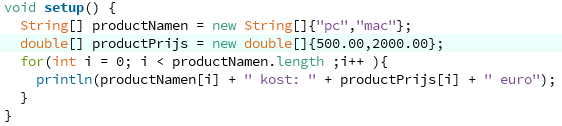
### Opgave A



Dit komt omdat de variable deLijjst geen referentie heeft naar een lijst. Dit komt omdat de referentie alleen aangekondigd is maar de daadwerkelijke pointer nog niet gezet is naar een object die wel bestaat.

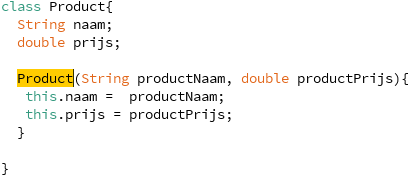
## Opgave Product

### Onderdeel A

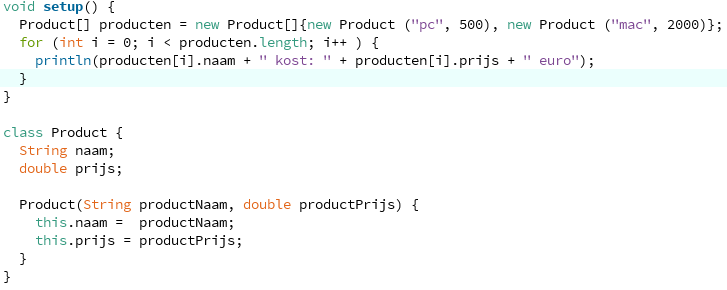


### Onderdeel B

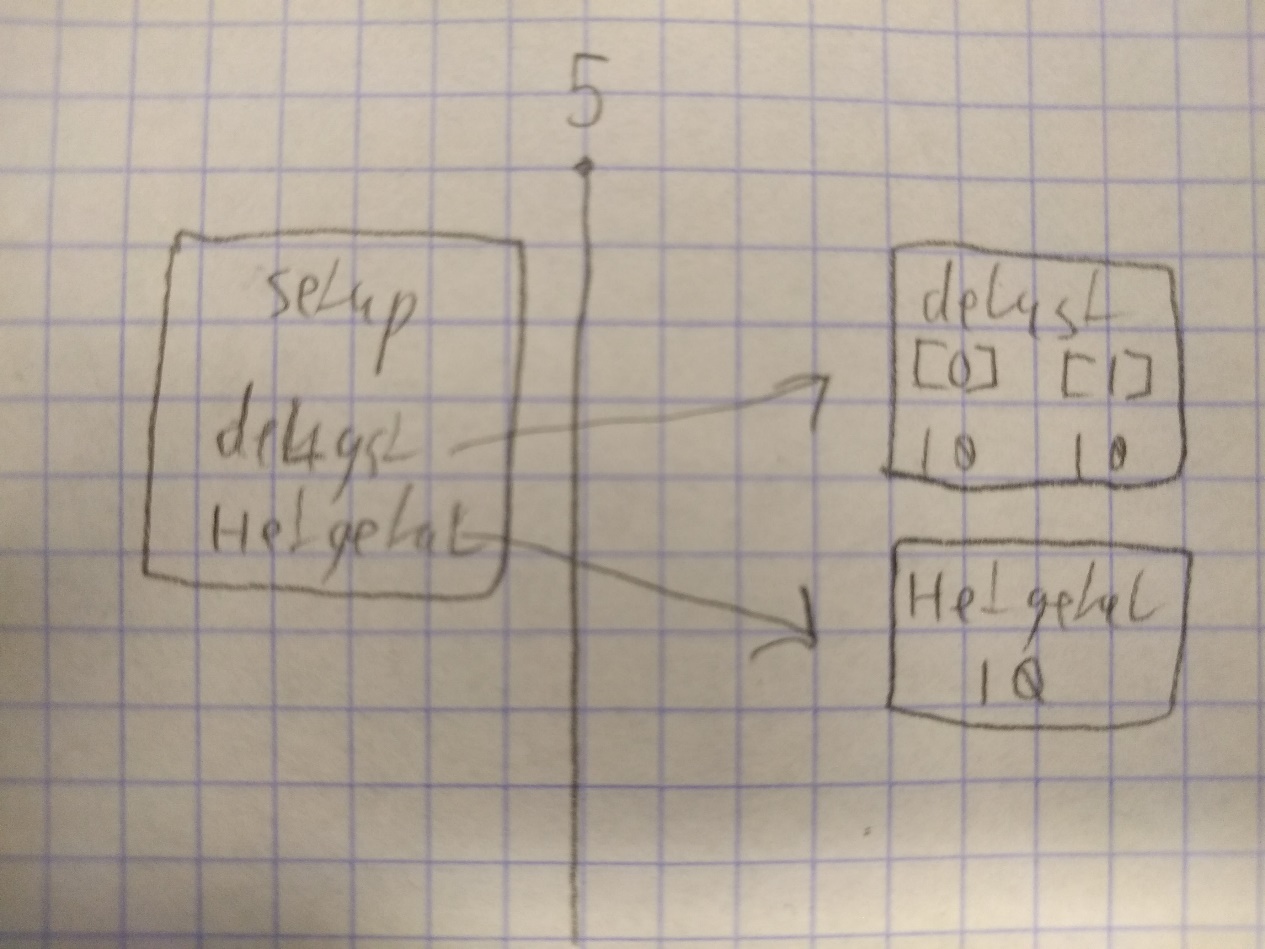
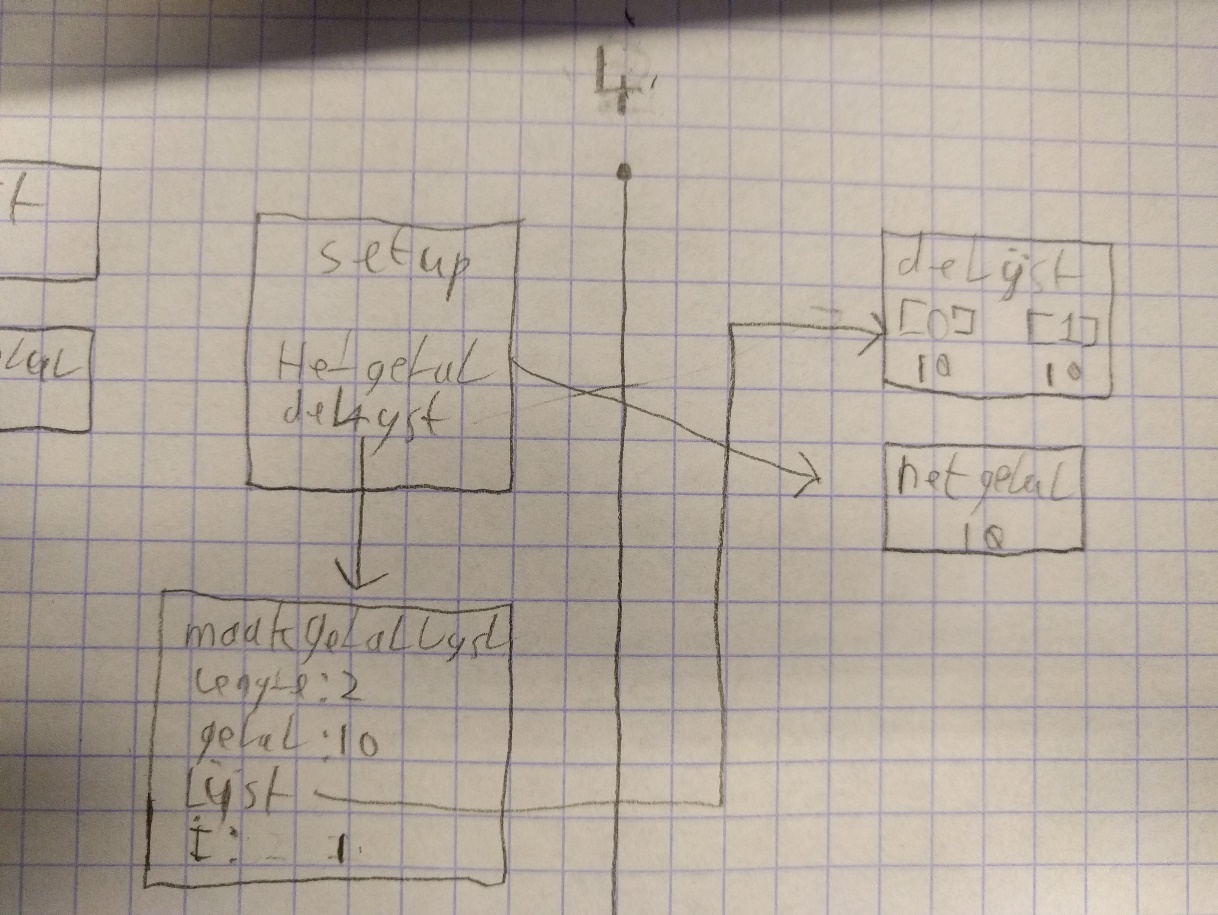
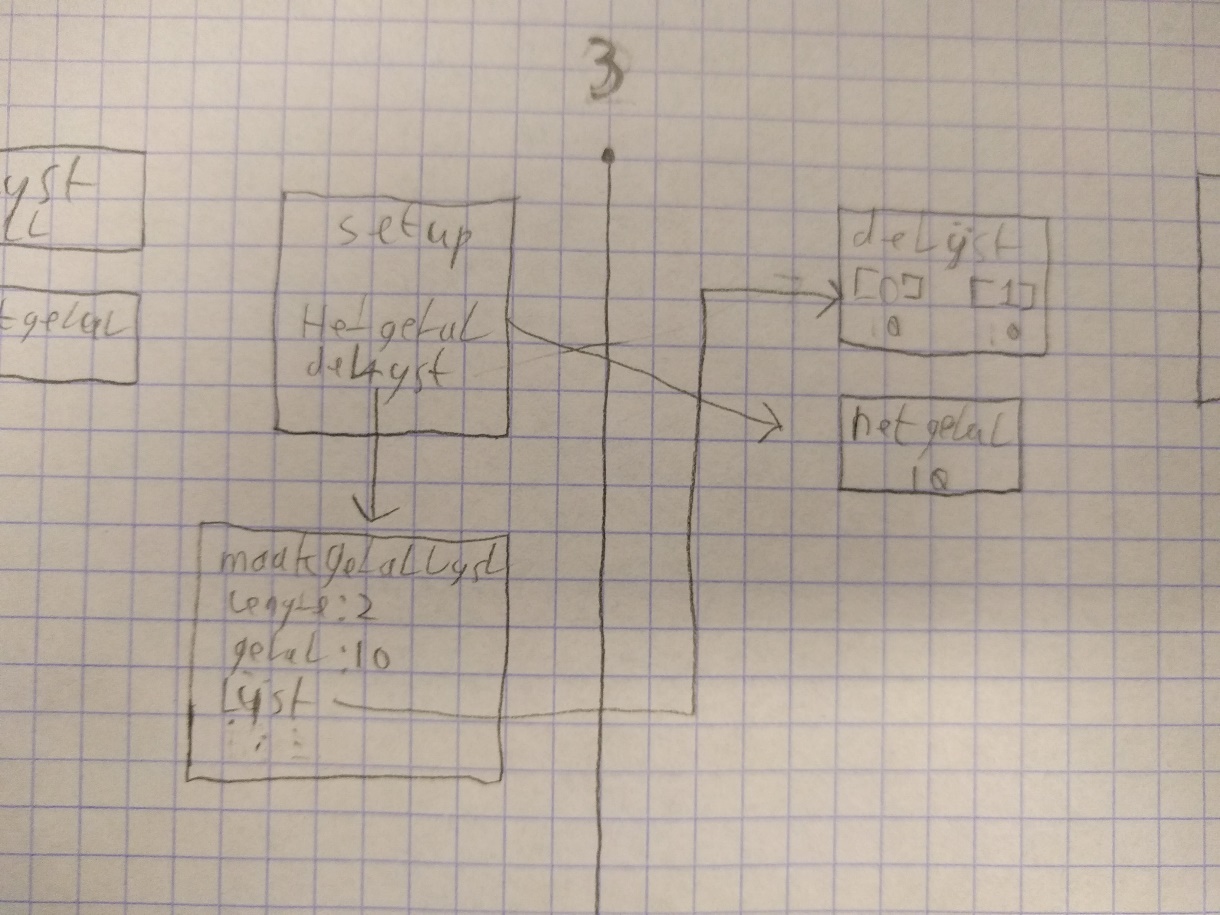
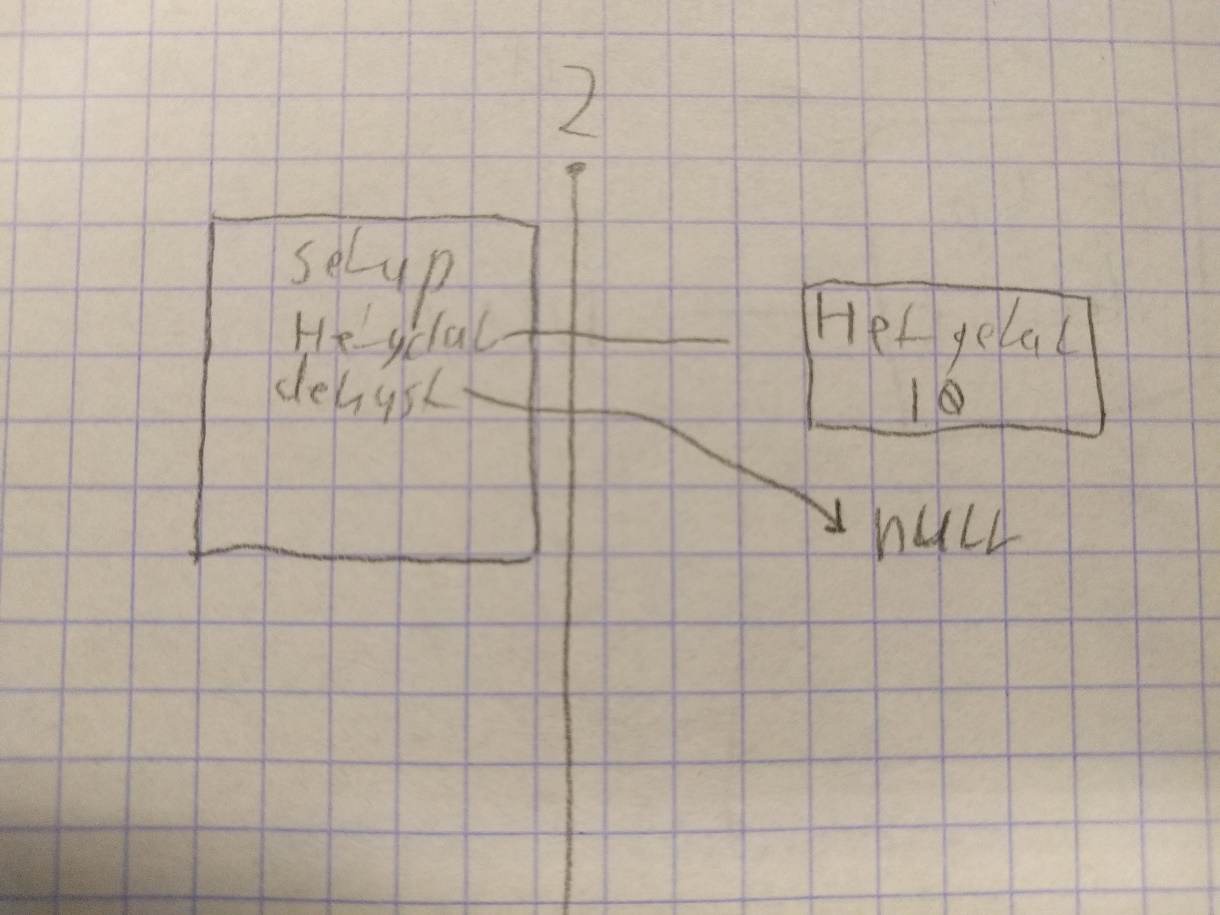
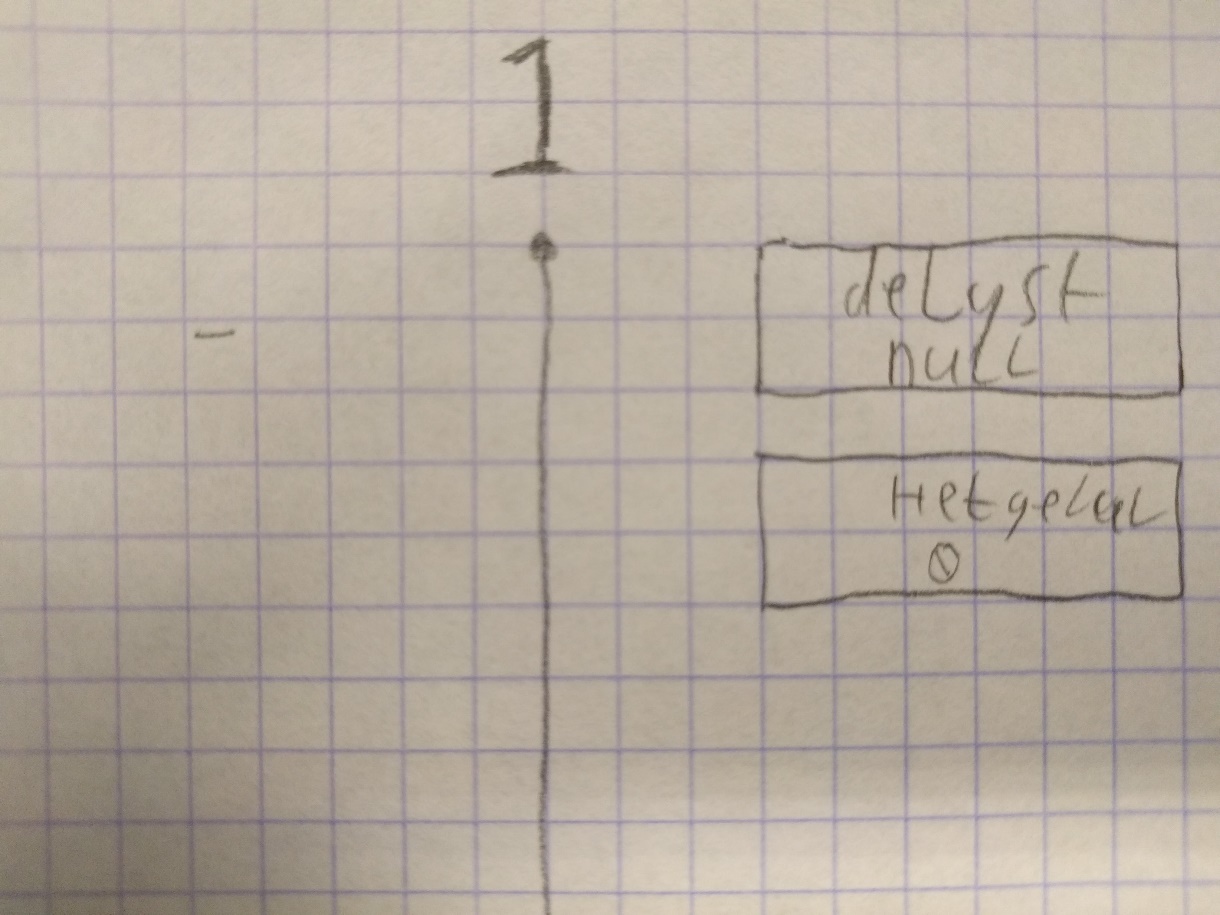
Het is allebei een member van een klasse. De product klasse b.v.b.



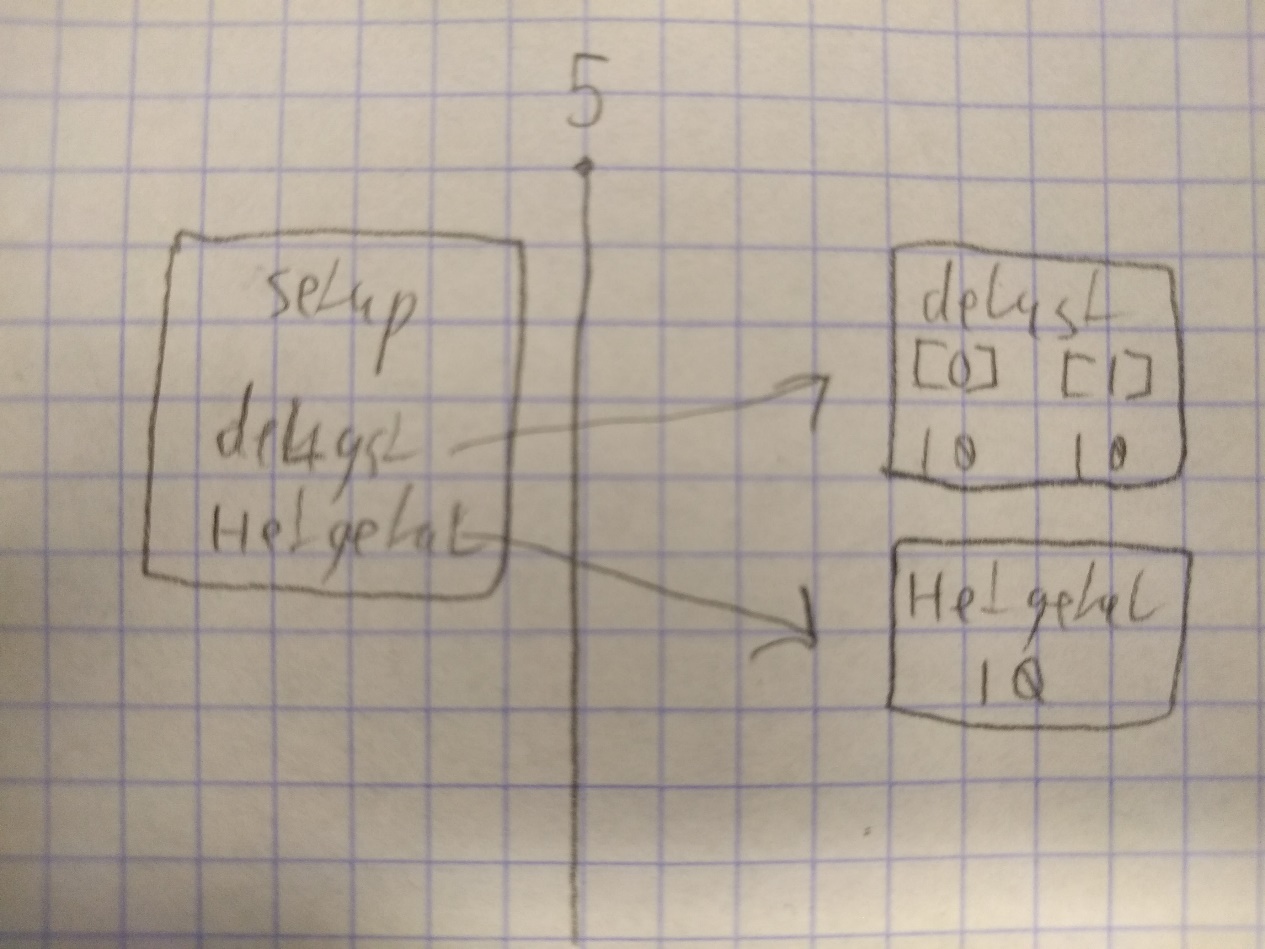
### Onderdeel C



# Oefeningen

Opgave geheugenmodel van doeFunctie, zie fotos hieronder.

### Onderdeel B



### Onderdeel C

Omdat de setup de lijst niet opbouw. Dat diet de methode ‘maakLijstMetEenGetal’.

De setup heeft dan geen referenties naar de members van het object “deLijst”

### Onderdeel D

Ik vind het niet handig omdat een functie zoals die niet duidelijk maakt dat hij gebruik maakt van globale variabelen. Als veel functies veel gebruik maken van globale variabelen dan verlies je ook controle over de wat de waardes representeren.