

Empowering Creative Thinking Through Programming

Want wat je hier ook komt studeren,
Je moet altijd programmeren.

Docenten (college)



Vincent Booman



Ton Markus

Docenten (Lab)



Vincent Booman



Ton Markus



Valentijn Muijrs



Aaron Oostdijk

Wat gaan we doen

Parachute/Speed Racer
(vangen/ontwijken)



Wat moet ik nou precies doen om dit vak te halen

- Mij absoluut NIET aanspreken met U.
- En al helemaal niet met meneer.

Wat moet ik nou precies doen om dit vak te halen

Lab/Eindopdracht

Iedere week is er lab/practicum.

Tijdens de labs bouw je verder aan je eindopdracht aan de hand van de les die je die week gehad hebt.

Je werk wordt iedere week beoordeeld en het gemiddelde van die beoordelingen vormt de beoordeling van je eindopdracht.

Wat moet ik nou precies doen om dit vak te halen

Lab/Eindopdracht + toets

Eindbeoordeling bestaat uit:

50% labs/eindopdracht

50% toets

Om het vak te halen moeten eindopdracht en toets ≥ 5.0
en moet het gemiddelde ≥ 5.5 zijn.

Eindopdracht is het gemiddelde van alle labs.

Wat moet ik nou precies doen om dit vak te halen

De Toets

Richting het eind van de lessen is er een toets waarbij er van je verwacht wordt dat je met behulp van het werk dat je tot op dat moment gedaan hebt ter plekke en individueel een programma kan maken.

In praktijk wil dat zeggen dat je je eindopdracht om moet kunnen bouwen tot iets dat er qua functionaliteit lijkt op je eindopdracht maar wel een ander spel is.

Wat moet ik nou precies doen om dit vak te halen

En hoe zit het dan met de lesopdrachten?

De lesopdrachten die je moet inleveren worden niet structureel beoordeeld.

Ze worden gebruikt als meta-informatie om je niveau te bepalen.

Wat is je niveau; kan je de lesopdracht en de lab opdracht?
Of, als de lab opdracht niet lukt, is de lesopdracht dan wel gelukt?

Waar gaan we het mee doen?

Processing (Java)

The image is a composite of two screenshots. The left screenshot shows the Processing website's homepage. It features a dark blue header with the 'Processing' logo and navigation links for 'p5.js', 'Processing.py', 'Processing for Android', and 'Processing for Pi'. Below the header is a sidebar with links to 'Cover', 'Download', 'Donate', 'Exhibition', 'Reference', 'Libraries', 'Tools', 'Environment', 'Tutorials', 'Examples', and 'Books'. The main content area includes a video titled 'Welcome to Processing 3!' from the Processing Foundation, a 'Donate' section, and a list of links to 'CreativeApplications', 'OpenProcessing', 'For Your Process', 'Processing Subr', 'Vimeo', and 'Studio Sketchpa'. The right screenshot shows the Processing IDE interface. The title bar reads 'sketch_190712a | Processing 3.5.3'. The interface includes a menu bar with 'File', 'Edit', 'Tools', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for running, stopping, and other functions. The main workspace is a large white area for writing code. At the bottom, there is a 'Console' and 'Errors' panel.

Processing p5.js Processing.py Processing for Android Processing for Pi

Processing

Cover
Download
Donate
Exhibition
Reference
Libraries
Tools
Environment
Tutorials
Examples
Books
Overview
People
» Forum
» GitHub
» Issues
» Wiki
» FAQ
» Twitter
» Facebook
» Medium

Welcome to Processing 3!
from Processing Foundation

22:03

» [Download Processing](#)
» [Browse Tutorials](#)
» [Visit the Reference](#)

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

» [CreativeApplications](#)
» [OpenProcessing](#)
» [For Your Process](#)
» [Processing Subr](#)
» [Vimeo](#)
» [Studio Sketchpa](#)

To contribute to the Processing community, please visit [Processing.org](#) to read instructions for [contributing code](#), [building from source](#), [reporting and tracking issues](#), and [creating libraries](#).

Partners

sketch_190712a | Processing 3.5.3

sketch_190712a

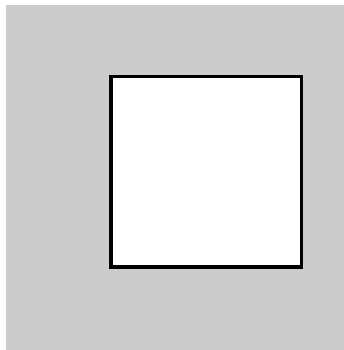
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Console Errors

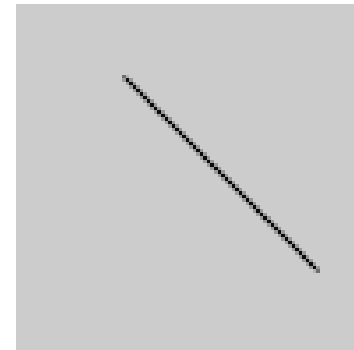
Wat doet Processing

Tekenen (20x per seconde)

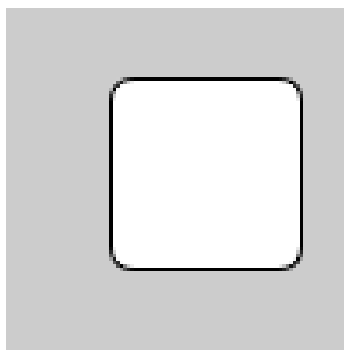
Tekenen op basis van cijfertjes



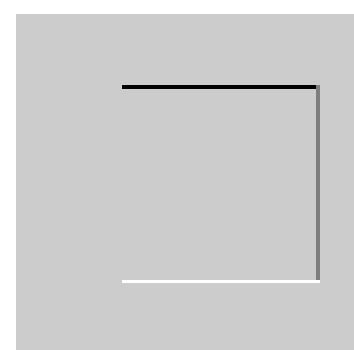
```
rect(30, 20, 55, 55);
```



```
line(30, 20, 85, 75);
```



```
rect(30, 20, 55, 55, 7);
```



```
line(30, 20, 85, 20);  
stroke(126);  
line(85, 20, 85, 75);  
stroke(255);  
line(85, 75, 30, 75);
```

Wat doet Processing

Maar voordat je kan gaan tekenen zal je eerst de cijfertjes moeten berekenen.

De truc is dat je met zo min mogelijk regels programmeer code een computer zo veel mogelijk laat rekenen/tekenen

Dat heet een programma schrijven
(Programmeren).

Programmeren doe je

Met een programmeertaal

In ons geval *Java*

Maar er zijn er meer (Python, C, C#, Swift ect)

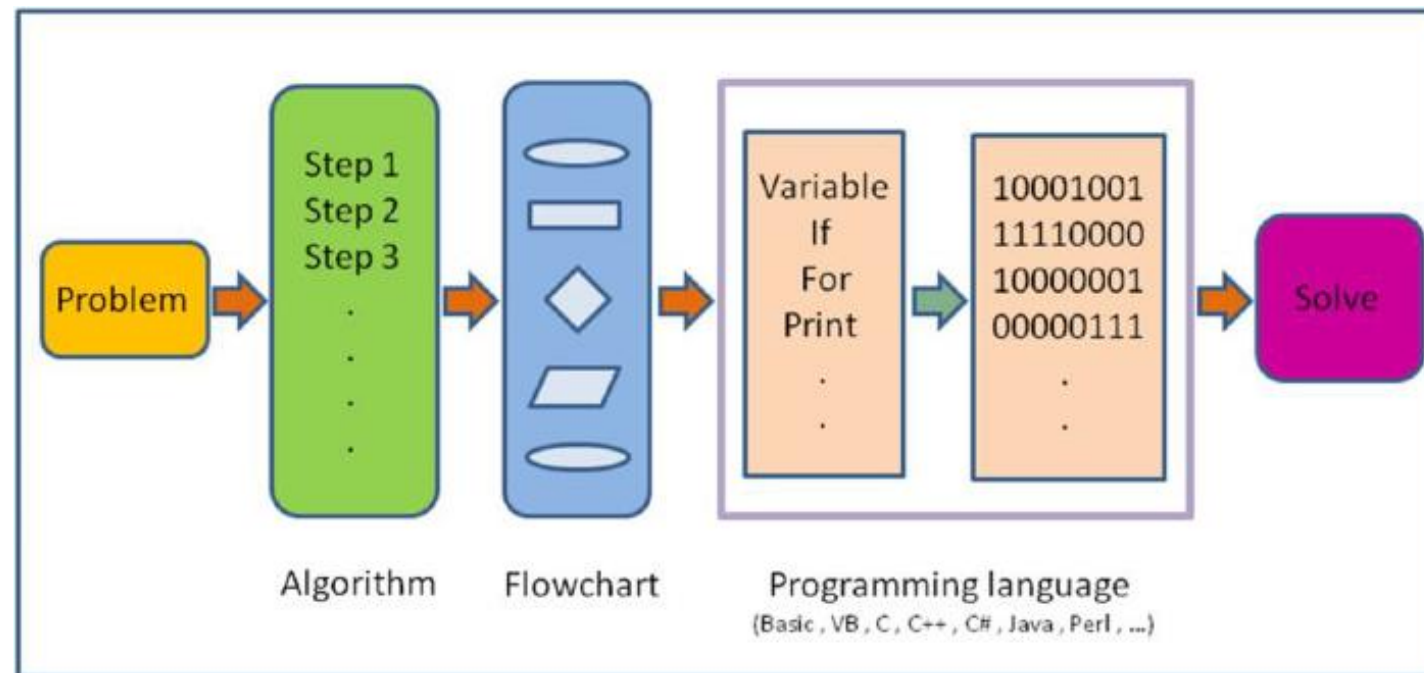
In een IDE

(Een programma om te programmeren)

In ons geval *Processing*

Maar er zijn er meer (Visual Studio, Xcode etc)

Programmeren is



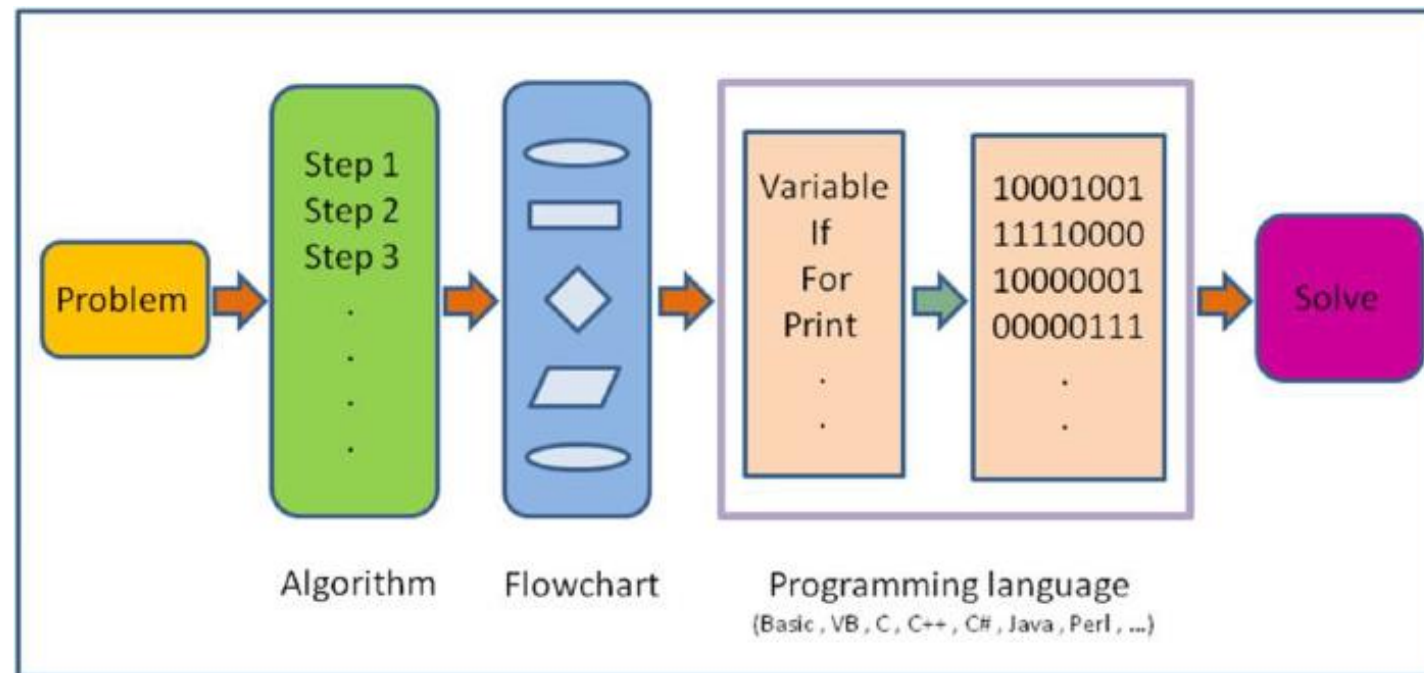
Geconfronteerd worden met een probleem

Het probleem goed kunnen formuleren

Creatief kunnen nadenken over oplossingen

Deze oplossingen duidelijk en accuraat weergeven

Programmeren is

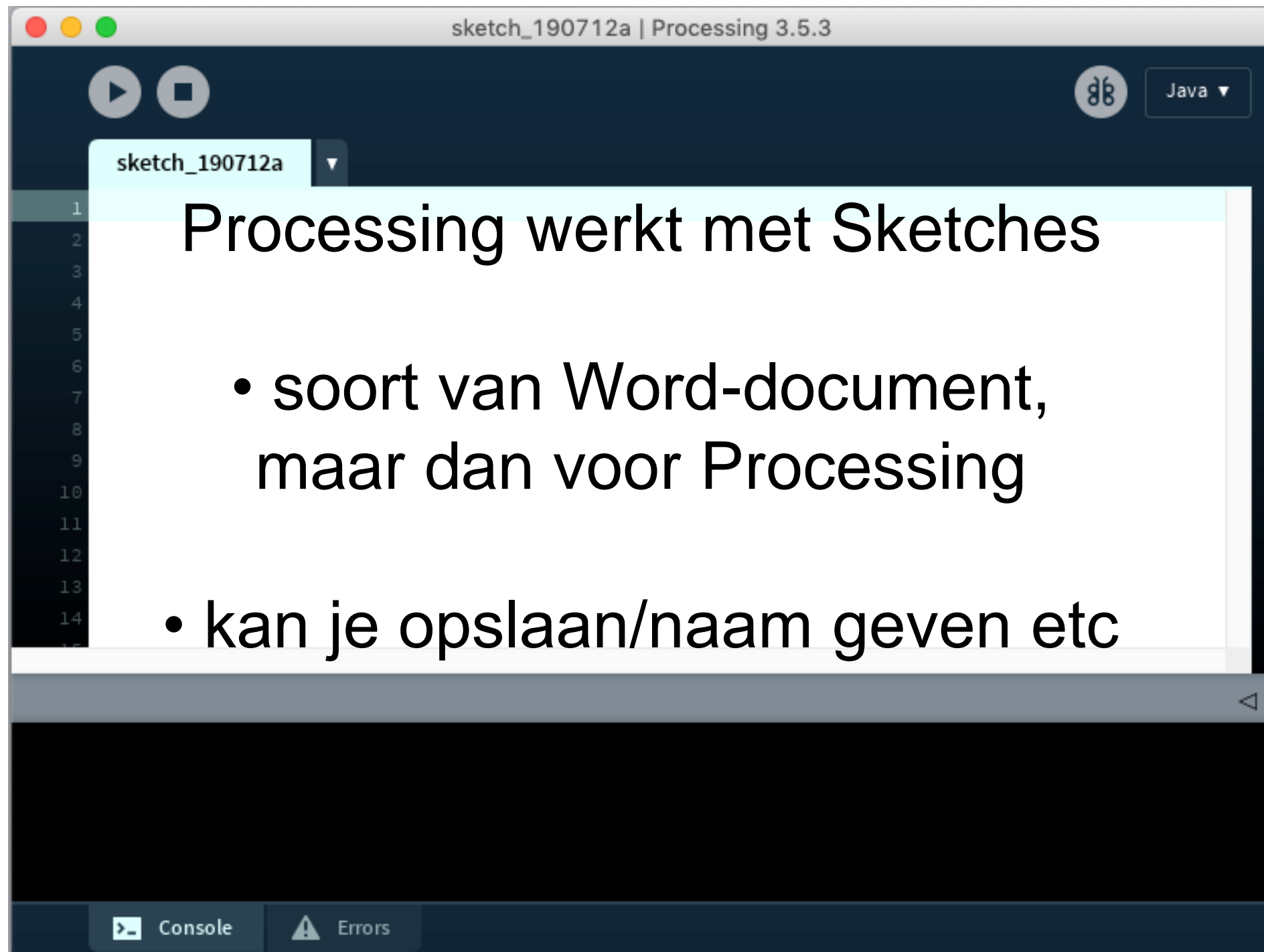


Zo'n oplossing heet een **algoritme**:

Een lijst met instructies die als ze exact uitgevoerd worden een oplossing voor het probleem vormen.







Lijst met instructies = lijst met programmeer regels

Hoe ziet dat er dan uit in Processing



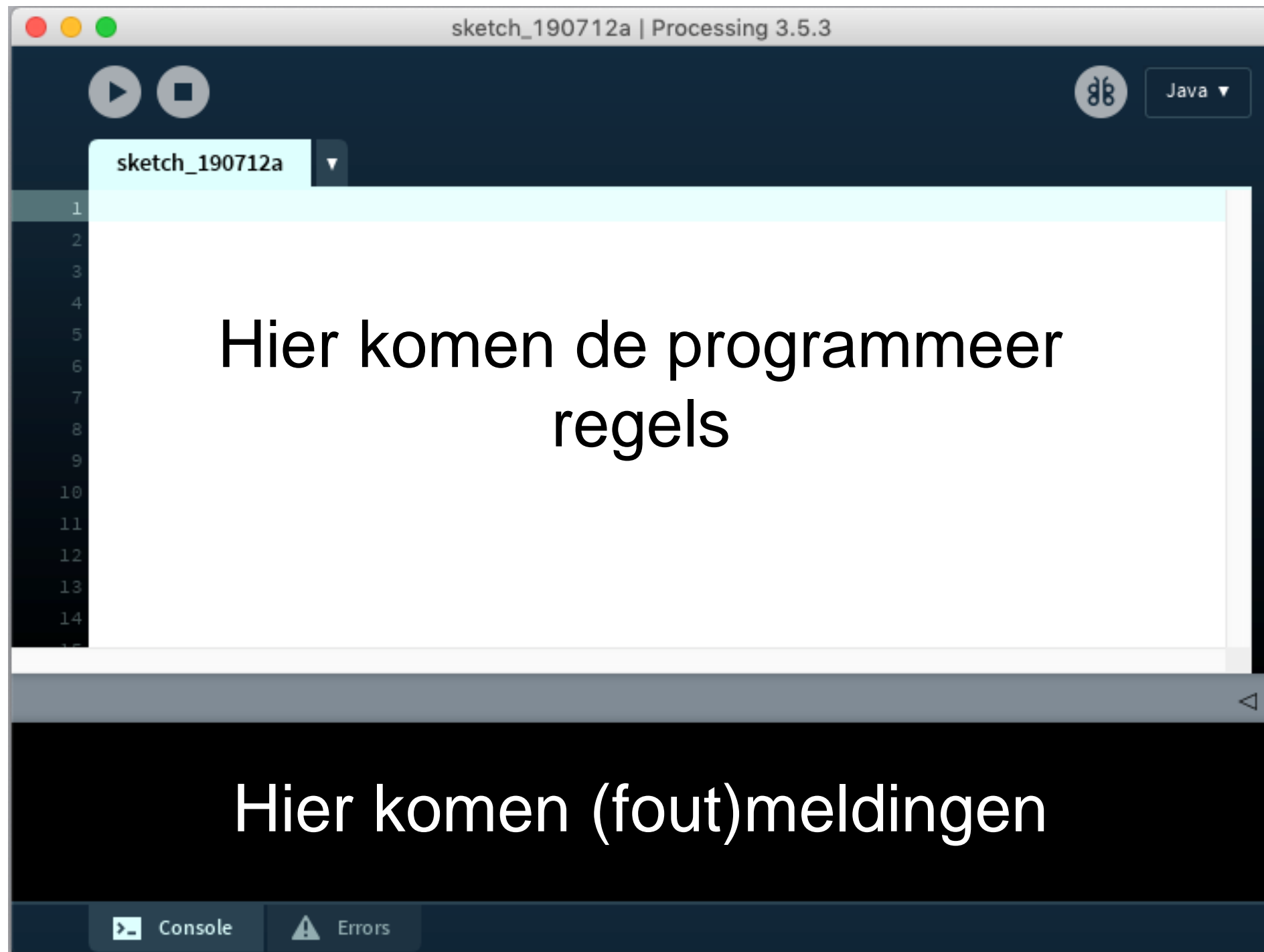
Bestanden

- Sketch is een .pde bestand
- .pde bestand staat in map met dezelfde naam als de sketch

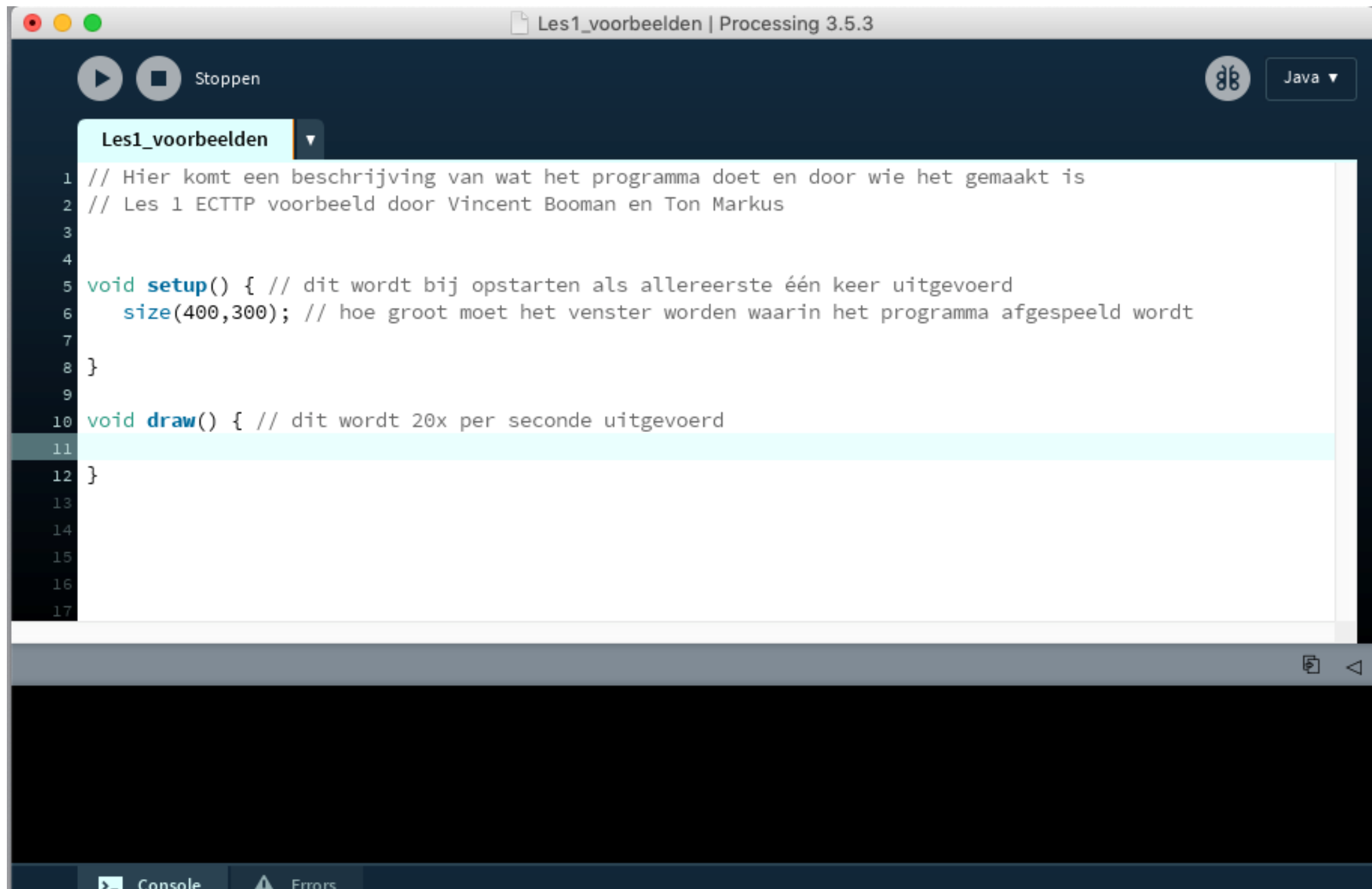
Vandaag	Vandaag
 Mijn1eSketch 	 Mijn1eSketch.pde 
 Schermafb...m 13.41.30 	

- Haal .pde bestand niet uit de map
- Bij inleveren huiswerk; map zippen, niet alleen het .pde bestand

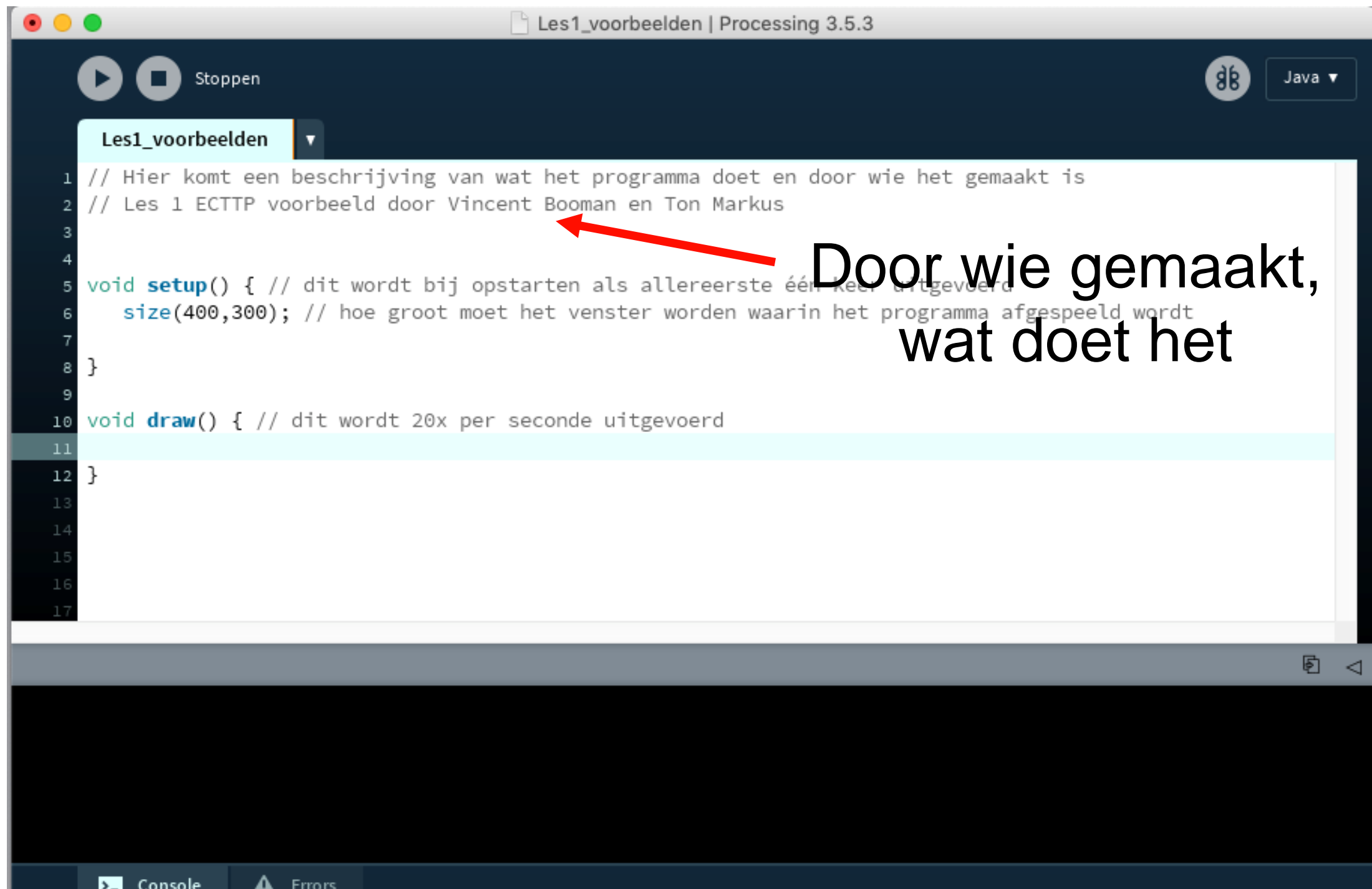
Hoe ziet dat er dan uit in Processing



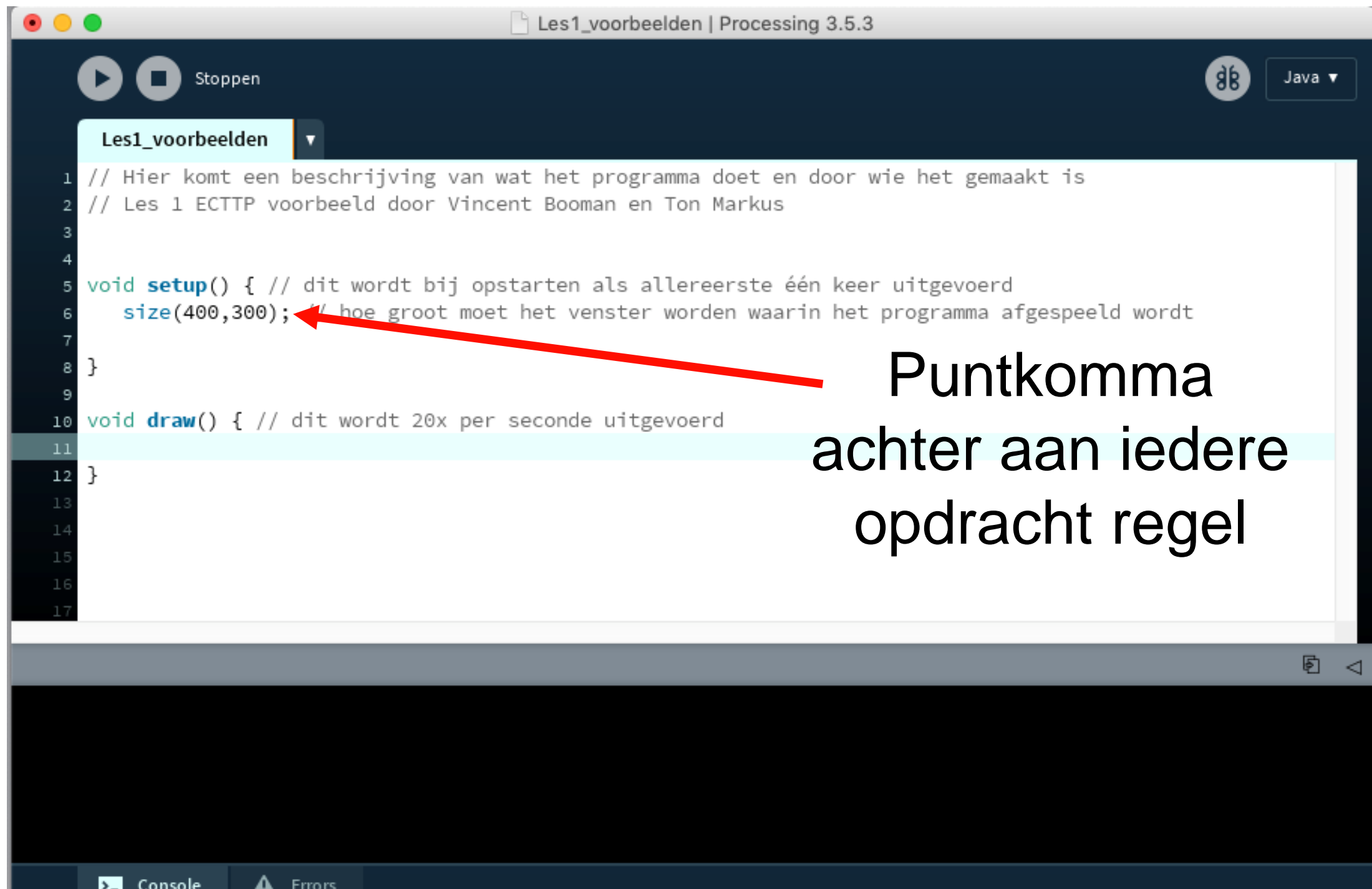
Wat moet er in ieder geval altijd in:



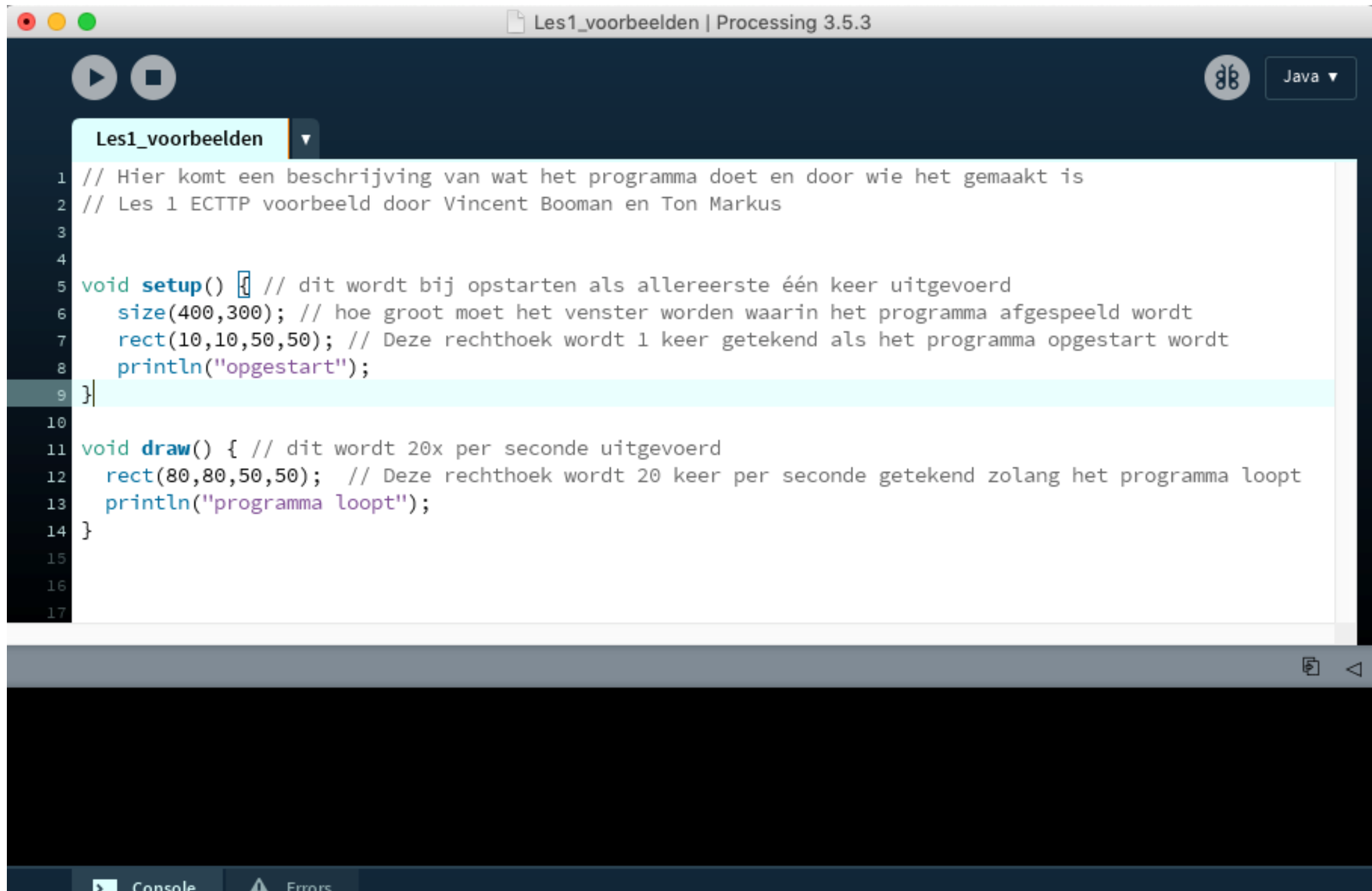
Wat moet er in ieder geval altijd in:



Wat moet er in ieder geval altijd in:



Je eerste tekenopdracht



The screenshot shows the Processing IDE interface. The title bar indicates the file is 'Les1_voorbeelden | Processing 3.5.3'. The code editor contains the following Java code:

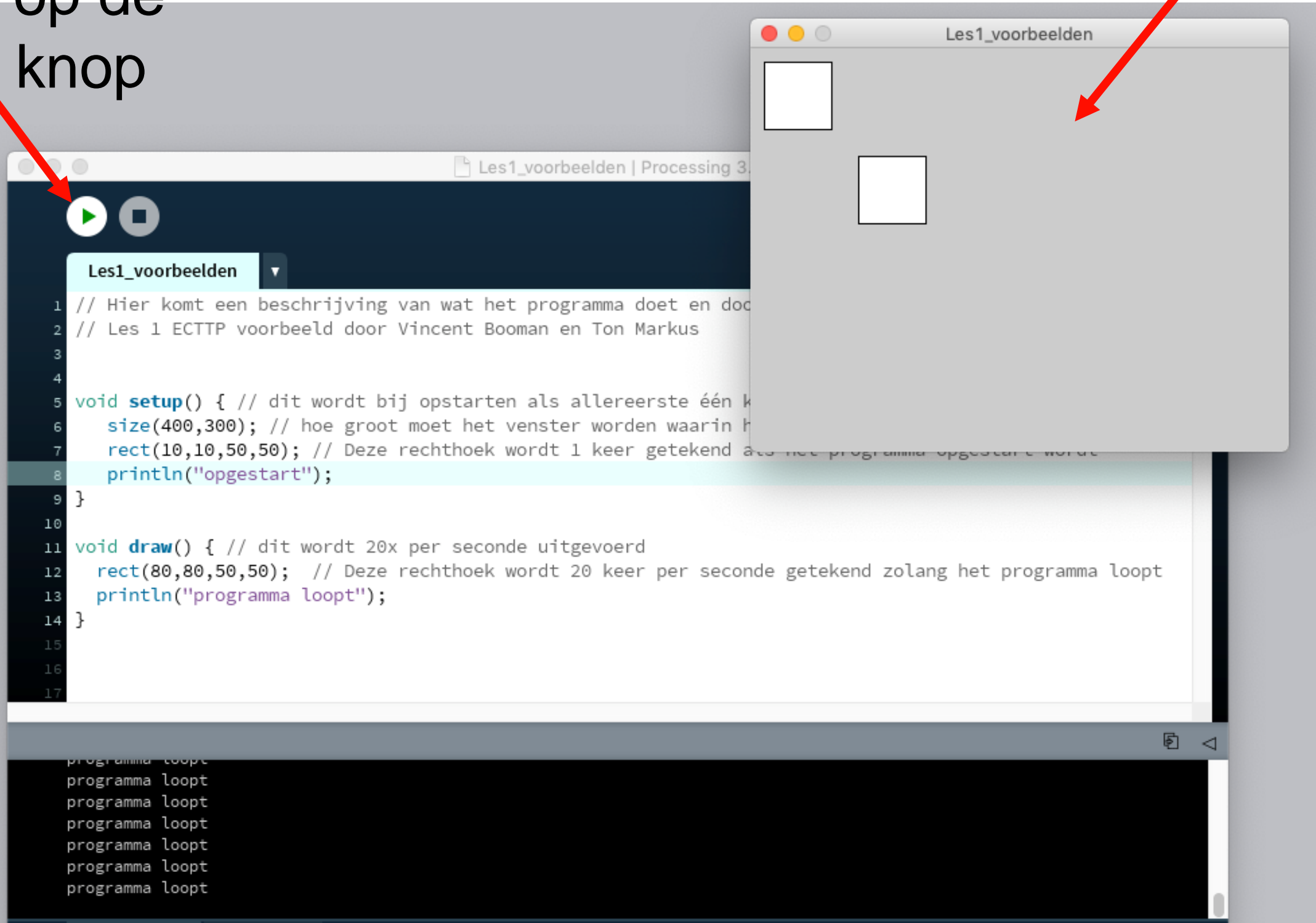
```
1 // Hier komt een beschrijving van wat het programma doet en door wie het gemaakt is
2 // Les 1 ECTTP voorbeeld door Vincent Booman en Ton Markus
3
4
5 void setup() { // dit wordt bij opstarten als allereerste één keer uitgevoerd
6   size(400,300); // hoe groot moet het venster worden waarin het programma afgespeeld wordt
7   rect(10,10,50,50); // Deze rechthoek wordt 1 keer getekend als het programma opgestart wordt
8   println("opgestart");
9 }
10
11 void draw() { // dit wordt 20x per seconde uitgevoerd
12   rect(80,80,50,50); // Deze rechthoek wordt 20 keer per seconde getekend zolang het programma loopt
13   println("programma loopt");
14 }
15
16
17
```

The IDE interface includes a toolbar with play and stop buttons, a language dropdown set to 'Java', and a bottom panel with tabs for 'Console' and 'Errors'.

Resultaat:

Hier wordt je
programma
'afgespeeld'

Druk op de
play knop



De cijfertjes uitgelegd:

Opdracht: teken
een rechthoek

Met deze cijfers

0,0

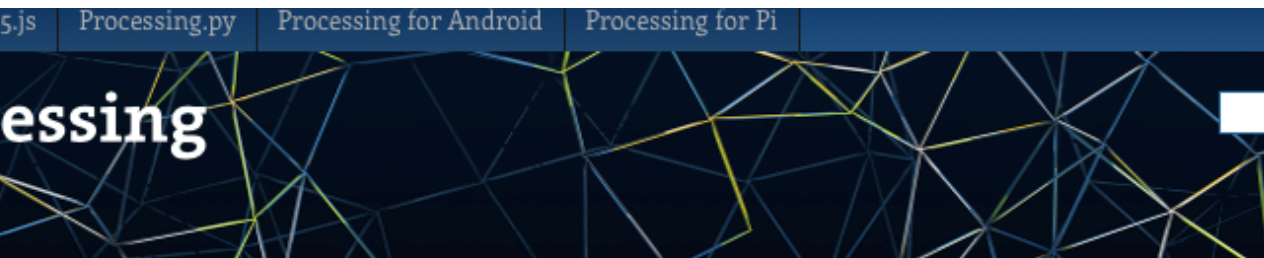
The screenshot shows a Processing IDE window titled 'Les1_voorbeelden'. The code editor contains the following code:

```
1 // Hier komt een beschrijving van wat het programma doet en doc
2 // Les 1 ECTTP voorbeeld door Vincent Booman en Ton Markus
3
4
5 void setup() { // dit wordt bij opstarten als allereerste één k
6   size(400,300); // hoe groot moet het venster worden waarin h
7   rect(10,10,50,50); // Deze rechthoek wordt 1 keer getekend als het programma opgestart wordt
8   println("opgestart");
9 }
10
11 void draw() { // dit wordt 20x per seconde uitgevoerd
12   rect(80,80,50,50); // Deze rechthoek wordt 20 keer per seconde getekend zolang het programma loopt
13   println("programma loopt");
14 }
```

The preview window shows a 400x300 canvas. A rectangle is drawn at (80, 80) with a width of 50 and a height of 50. Red arrows indicate the dimensions: 80 (1e) x 50 (1e) for the first rectangle and 80 (2e) y 50 (2e) hoogte for the second rectangle. The text 'Breedte' is also present.

400,300

‘alle’ opdrachten



Reference. Processing was designed to be a flexible software sketchbook.

Structure

- () (parentheses)
- , (comma)
- . (dot)
- /* */ (multiline comment)
- /** */ (doc comment)
- // (comment)
- ;(semicolon)
- = (assign)
- [] (array access)
- { } (curly braces)
- catch
- class
- draw()
- exit()
- extends
- false
- final
- implements
- import
- loop()
- new
- noLoop()
- null
- pop()
- popStyle()
- private
- public
- push()
- pushStyle()
- redraw()
- return

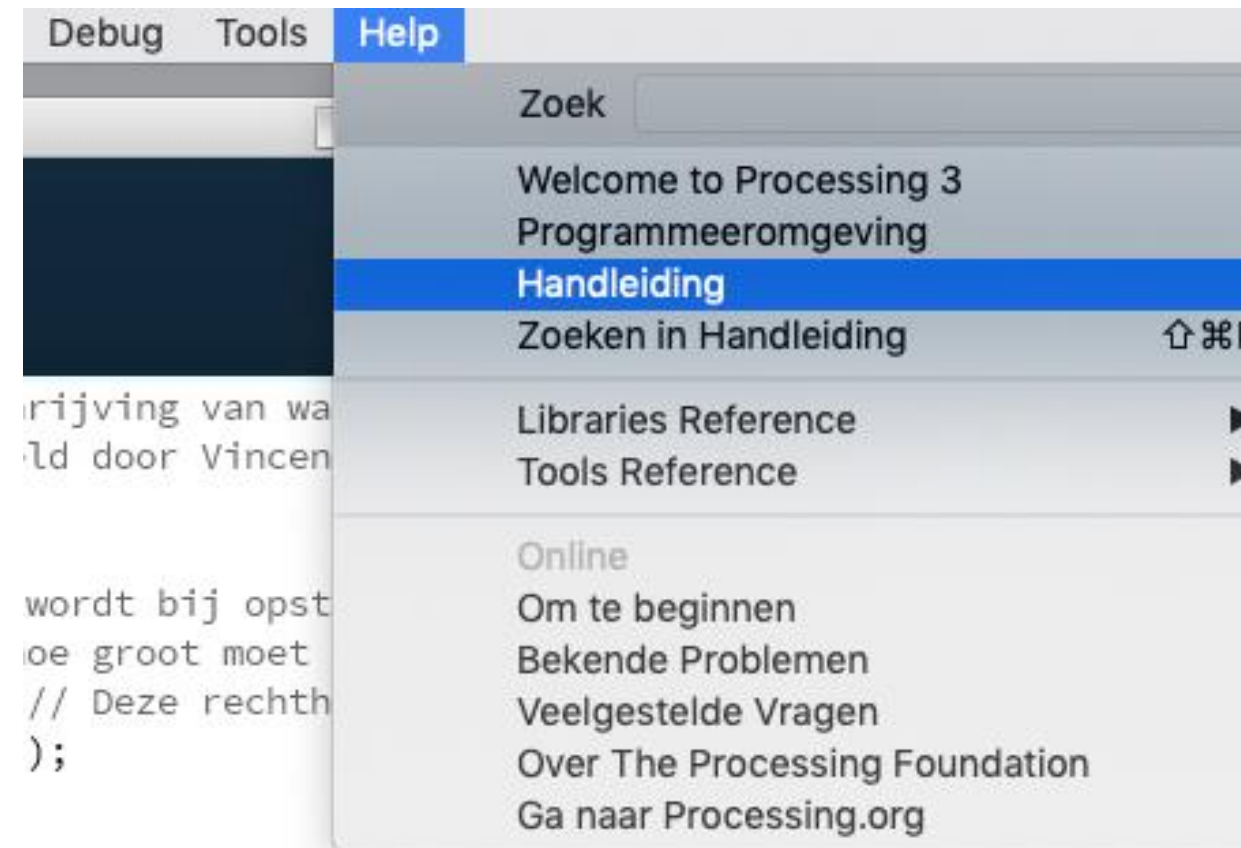
Shape

- createShape()
- loadShape()
- PShape
- 2D Primitives
 - arc()
 - circle()
 - ellipse()
 - line()
 - point()
 - quad()
 - rect()
 - square()
 - triangle()
- Curves
 - bezier()
 - bezierDetail()
 - bezierPoint()
 - bezierTangent()
 - curve()
 - curveDetail()
 - curvePoint()
 - curveTangent()
 - curveTightness()
- 3D Primitives
 - box()
 - sphere()
 - sphereDetail()

Color

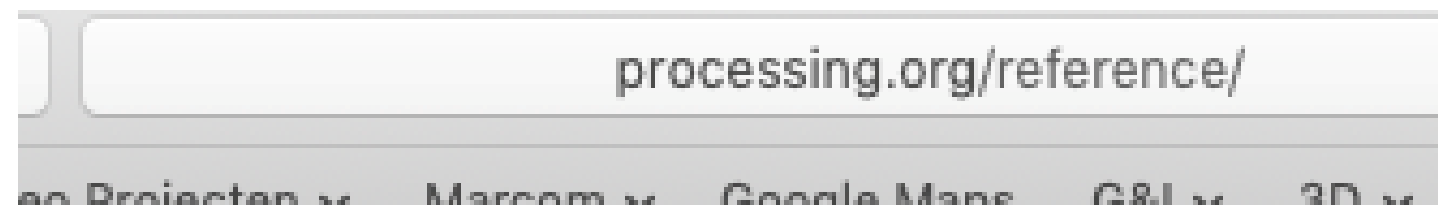
- Setting
 - background()
 - clear()
 - colorMode()
 - fill()
 - noFill()
 - noStroke()
 - stroke()
- Creating & Reading
 - alpha()
 - blue()
 - brightness()
 - color()
 - green()
 - hue()
 - lerpColor()
 - red()
 - saturation()
- Image
 - createImage()
 - PImage
- Loading & Displaying
 - image()
 - imageMode()
 - loadImage()

Via:

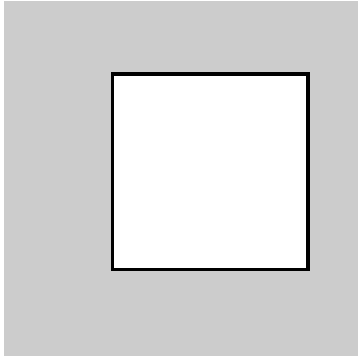
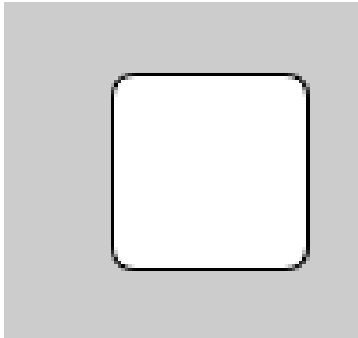


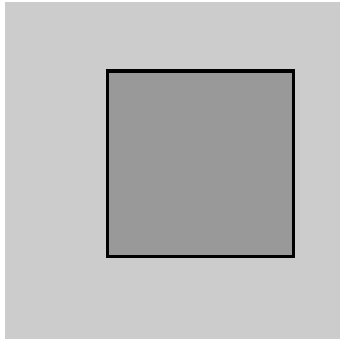
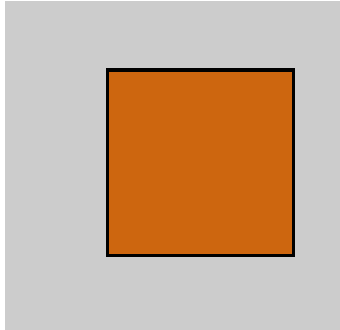
```
wordt 20x per seconde uitgevoerd
// Deze rechthoek wordt 20 keer per seconde getekend zodat
  point() :
```

of:



Welke gaan we gebruiken:

Name	rect()
Examples	 <pre>rect(30, 20, 55, 55);</pre>
	 <pre>rect(30, 20, 55, 55, 7);</pre>

Name	fill()
Examples	 <pre>fill(153); rect(30, 20, 55, 55);</pre>
	 <pre>fill(204, 102, 0); rect(30, 20, 55, 55);</pre>

Lesopdracht 1

(dit is niet de huiswerk/lab-opdracht)

- Download Processing via <https://processing.org/download>
- Let op wat voor platform je hebt
- Let op uitschakelen virus protection voor Processing (Processing doet dingen waarvan Windows kan denken dat het een virus is)

Lesopdracht 2

(dit is ook niet de huiswerk/lab-opdracht)

- teken meerdere rechthoeken en ovalen van verschillende groottes en verschillende kleuren
 - zorg dat minimaal 1 rechthoek en 1 ovaal precies tegen elkaar aan liggen.
 - inleveren via <https://gi1.hku.nl>
- (inloggen met je mailadres en het wachtwoord dat je bij je mailadres hebt gekregen)

Meer informatie?

- <https://github.com/ecttp>