SPD MODULE 7: ARRAYS

LESSTOF

Screencasts: https://www.youtube.com/playlist?list=PLA42824F8B7E248BD

Reader: Hoofdstuk 7

OEFENOPGAVEN

OPGAVE 7.1

Declareer een globale array van Strings en zet daar de volgende boodschappen in; Brood, Melk, Eieren, Vleeswaren en Koekjes.

Druk deze boodschappen af in de console. Processing accepteert arrays in de *println()* en er is een methode *printArray()*, maar het gaat er nu juist om dat je dat zelf programmeert.

OPGAVE 7.2

Waarschijnlijk heb je in de vorige methode gewoon in de *setup()* geprogrammeerd of de passive mode gebruikt (zonder methoden). Maak nu een methode die als parameter een String-array krijgt en alle waardes één voor één afdrukt. Gebruik de volgende code als basis:

```
String[] boodschappen = { .... };

void setup() {
  drukAf( boodschappen );
}

// Hier jouw 'drukAf()'-methode met één parameter (String-array)
```

OPGAVE 7.3

Hiernaast staat de top 5 jongens- en meisjesnamen van 2015[1]. Maak een programma waarmee je die top 5 per sekse afdrukt. Maak twee arrays met de namen en in ieder geval een methode om een top 5 af te drukken. De uitvoer ziet er zo uit:

Bron: [1] https://www.svb.nl/int/nl/kindernamen/artikelen/top20/archief/2015/

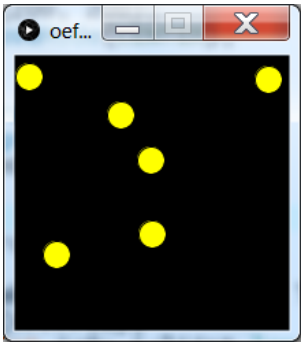
OPGAVE 7.4

Gegeven is onderstaande code, de opdrachten zijn gebaseerd op deze code. De code begint met een tweedimensionale array met coördinaten (middelpunten van cirkels). In dit voorbeeld is een lijst van 6 coördinaten gegeven, de eerste dimensie bevat de x-coördinaat, de tweede de y-coördinaat. Dus het element `cirkels[2][0]` bevat de x-coördinaat van de cirkel en `cirkels[2][1]` de y-coördinaat.

```
// Een array van int[6][2] met coördinaten van cirkels
int[][] cirkels = { {10,15},{100,130},{77,43},{30, 145},{185,17},{99,76} };
final int DIAMETER = 20;
final int GEEL = #FFFF00;
final int ROOD = #FF0000;

void setup() {
  size(200, 200);
  ellipseMode(CENTER);
}

void draw() {
  background(#000000);
  tekenCirkels(cirkels);
}

// Hieronder jouw code
```

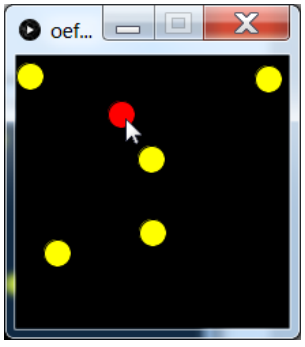
OPDRACHT 7.4 A

Voeg aan de bovenstaande code de methode *tekenCirkels()* toe, zodat alle cirkels getekend worden (je mag de bovenstaande code *niet* wijzigen, alleen nieuwe methoden toevoegen). Het scherm ziet er als volgt uit:



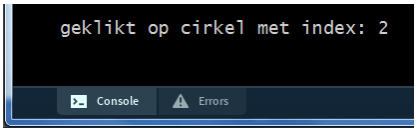
OPDRACHT 7.4 B

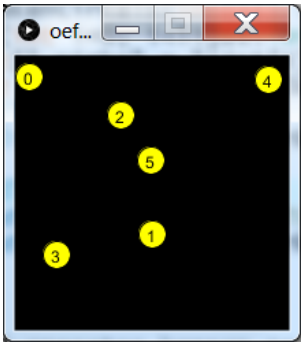
Wijzig je methode *tekenCirkels()* dat als je met de muis boven een cirkel gaat staan, deze rood gekleurd wordt. Ziet er als volgt uit:



OPDRACHT 7.4 C

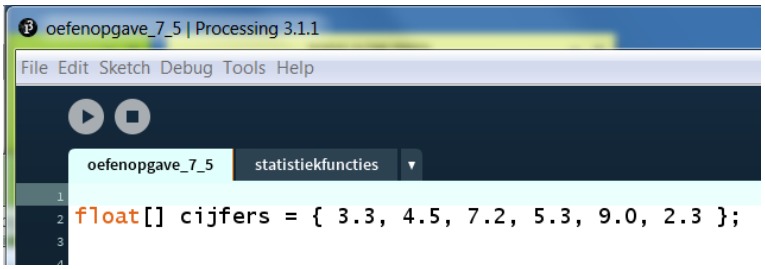
Als je nu klikt op een cirkel, druk dan in de tekstconsole de positie van die cirkel in de array af. Bij het voorbeeld van opdracht 7.5 is op de derde cirkel (index 2 dus!) geklikt. De uitvoer is dan: (hiernaast alle cirkels met de positie in de array).





OPGAVE 7.5

Voor de moduleopgave heb je een aantal statistische functies nodig. Een aantal van deze functies ga je nu maken. Zet deze functies in een apart tabblad 'statistiekfuncties', zodat je 'm eenvoudig kunt hergebruiken bij de moduleopgave.



OPDRACHT 7.5 A

Gebruik de volgende code (op het eerste blad):

```
float[] cijfers = { 3.3, 4.5, 7.2, 5.3, 9.0, 2.3 }; // min 2.3, max 9.0

void setup() {
  float hoogste = geefHoogste(cijfers);
  println("Het hoogste cijfer is: " + hoogste);
}
```

Maak op het tabblad ´statistiekfuncties´ de methode *geefHoogste()* waar je een float-array aan meegeeft en die de hoogste waarde uit die lijst teruggeeft. Let op, Processing heeft een standaard *max()* functie. Het is niet de bedoeling die te gebruiken, maar dat je zelf de hoogste waarde uit de array haalt door er doorheen te lopen.

Klopt je uitkomst?

OPDRACHT 7.5 B

Test je methode *geefHoogste()* met de volgende twee arrays:

```
float[] lijst1 = { -8.0, 4.5, 1.2, -5.6, -9.1, 4 }; // min -9.1, max 4.5
float[] lijst2 = { -0.4, -0.7, -3.5, -1.9, -8.0 };  // min -8.0, max -0.4
```

Kloppen je uitkomsten? Zo nee, dan is je methode niet correct. Pas deze aan zodat het wel klopt (voor elke willekeurige float-array). En test met de drie gegeven arrays.

OPDRACHT 7.5 B

Voeg de volgende code toe aan je *setup()*:

```
float laagste = geefLaagste(cijfers);
println("Het laagste cijfer is: " + laagste);
```

En implementeer de methode *geefLaagste()* in het tabblad statistiekfuncties (ook hier, niet de *min()* functie gebruiken).

Klopt je uitkomst? Heb je dat met alle drie de arrays getest?

MODULEOPGAVE 7: CIJFEROVERZICHT

PROBLEEMSTELLING

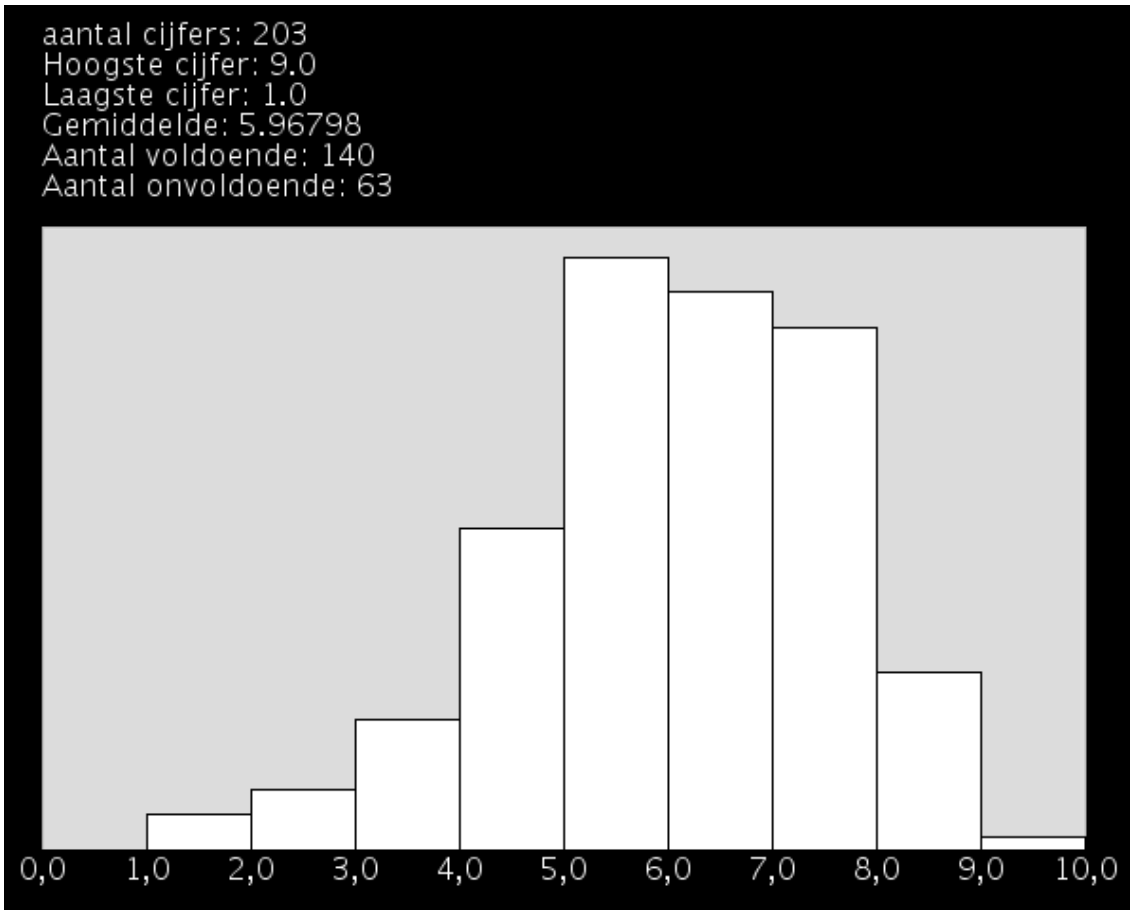
Het docententeam van de course Web Development wil analyseren hoe de afgelopen jaren toetsen gemaakt zijn. De trekker van de course heeft alle behaalde cijfers van het afgelopen jaar verzameld in een tekstbestand en wil daar een aantal analyses op los laten.

Het team heeft behoefte aan de volgende gegevens:

1. het aantal toetsen,
2. het hoogst behaalde cijfer,
3. het laagst behaalde cijfer,
4. het gemiddelde cijfer,
5. het aantal voldoendes (>= 5,5)
6. het aantal onvoldoendes (<5,5)

Verder wil het team een grafiek zien met de verdeling van de cijfers. Daar wordt mee bedoeld dat er een staafdiagram getekend moet worden van het aantal cijfers tussen 0 en 1, 1 en 2, 3 en 4, etc. Dus de eerste staaf representeert het aantal cijfers waarbij geldt >=0 en =<1, de tweede staaf >1 en <=2, etc.

Zie hieronder voor een afbeelding (gebaseerd op het bestand ‘cijfers_groot.txt’).



OPDRACHT

Maak een programma dat bovenstaande kan. Maak je programma eerst met het bestand ‘cijfers_klein.txt’ (dat is een klein deel van de actuele cijfers). Als alles werkt, test je programma dan met het bestand ‘cijfers_groot.txt’. In beide situaties moet je programma correct werken, bijv. de grafiek moet in zijn geheel zichtbaar zijn (dus schalen naar aantallen).

Volg het volgende stappenplan:

1. Maak een nieuw project in Processing en voeg het tabblad ‘statistiekfuncties’ van Opgave 7.5 toe. Breid dat tabblad uit met alle statistiekfuncties die gevraagd zijn (zie screenshot hiervoor). Maak alle methode helemaal zelf, dus gebruik geen standaardfuncties van Processing (zoals *min()* of *max()*). Uiteindelijk heb je 6 methoden: ‘geefAantal’, ‘geefHoogste’, ‘geefLaagste’, ‘geefGemiddelde’, ‘geefAantalVoldoendes’ en ‘geefAantalOnvoldoendes’. Test je methoden voordat je de volgende stappen uitvoert.
2. Met de methode *loadStrings()* van Processing kun je een tekstbestand regel voor regel inlezen. Maak een eigen methode die het bestand inleest en omzet naar een array met cijfers en die returnt.
3. Bepaal eerst de statistieken en druk die linksboven af op het scherm. Maak per statistiek een aparte methode en maak hiervoor **geen** gebruik van de standaard Processing functies max() en min()!
4. Bepaal per “range” het aantal behaalde cijfers, de ranges zijn (als tussenstap druk de resultaten eerst af in de textarea met println).
    - cijfer <= 1
    - 1 < cijfer <= 2
    - 2 < cijfer <= 3
    - 3 < cijfer <= 4
    - 4 < cijfer <= 5
    - 5 < cijfer <= 6
    - 6 < cijfer <= 7
    - 7 < cijfer <= 8
    - 8 < cijfer <= 9
    - 9 < cijfer <= 10
5. Teken het staafdiagram, zorg dat de staven altijd “zo groot mogelijk” op je scherm passen.

EXTRA OPGAVEN

EXTRA OPGAVE 7.1: CASUS GELDWISSELAPPARAAT

In deze opgave ga je een algoritme maken dat een bepaald bedrag kan opdelen in het aantal muntstukken dat je nodig hebt om tot dat bedrag te komen.

PROBLEEMBESCHRIJVING

De invoer is het bedrag dat opgedeeld moet worden. De uitvoer een plaatje waarin het aantal munten van elke soort is getekend (zie screenshots). Gebruik in je programma een array om alle munten van de gebruikte muntsoort op te slaan. Je moet bijvoorbeeld tussen euro's en guldens kunnen wisselen door enkel de array aan te passen. Om afrondingsproblemen met float te voorkomen, kun je het beste van centen uitgaan voor de muntwaarde. Dus 1 euro is 100 centen.

Een systematische manier om te bepalen welke muntstukken je moet teruggeven om tot een bepaald bedrag te komen, is de volgende:

Je deelt het bedrag dat je nog moet teruggeven door het muntstuk met de hoogste waarde dat deelbaar is door het terug te geven bedrag. De uitkomst bepaalt hoeveel muntstukken je van deze soort nodig hebt. Vervolgens trek je de totale waarde van deze muntstukken af van het bedrag dat je nog moet opdelen in munten. Dit proces herhaal je totdat het bedrag dat je nog moet opdelen op 0 uitkomt.

OPGAVE A

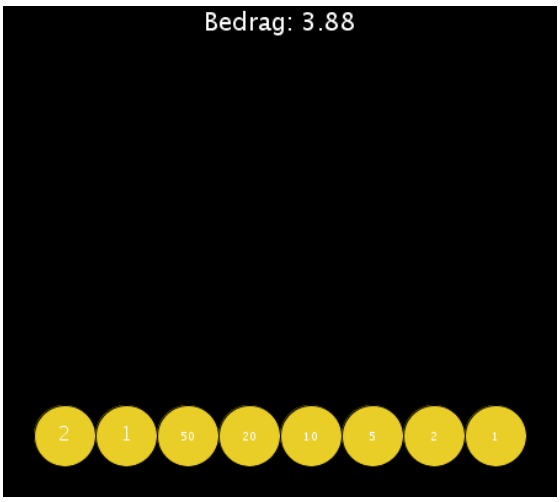
Maak eerst een oplossing die met println in de textarea afdrukt hoeveel munten van elke waarde teruggegeven moeten worden. Bijvoorbeeld zoiets als in Figuur 1 en Figuur 2:

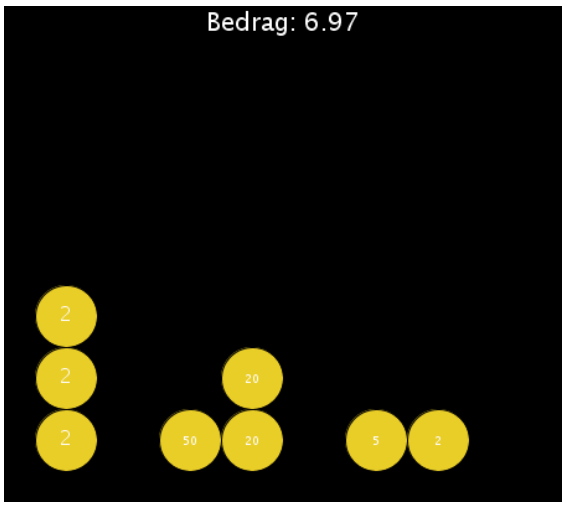
Figuur 1Bedrag in euro's

Figuur 2Bedrag in gulden

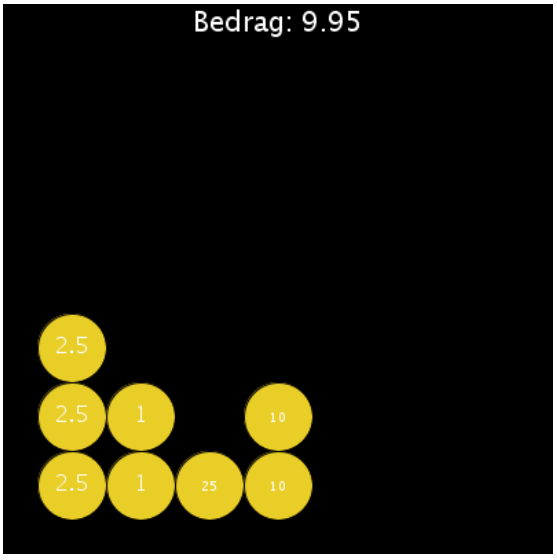
OPGAVE B

Maak nu een visuele representatie van de uitkomst. Hieronder zie je een aantal voorbeelden.





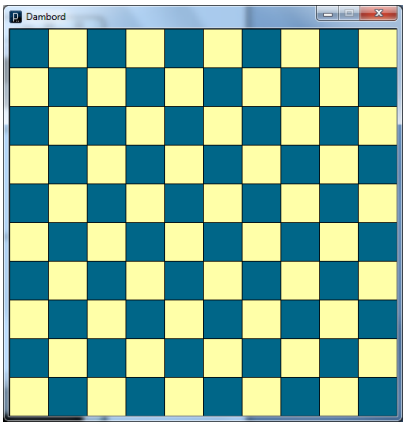
Voorbeelden van een bedrag verdeeld in euromunten



Voorbeeld van een bedrag verdeeld in guldenmunten.

EXTRA OPGAVE 7.2

Teken in Processing een dambord waarvan de grootte van de vakken en damstenen automatisch mee schaalt met de grootte van het bord.



OPGAVE 7.2

Teken de damstenen in het dambord (zie onderstaand voorbeeld). Zorg hierbij ervoor dat het tekenen ook werkt als posities van bepaalde damstenen zijn veranderd (zie ander voorbeeld), dus denk goed na over hoe de informatie over de positie van de damstenen wordt bijgehouden.