SPD MODULE 5: RETURNWAARDEN

LESSTOF

Screencasts: https://www.youtube.com/playlist?list=PL802B4F374EF5A5F6

Reader: Hoofdstuk 5

OEFENOPGAVEN

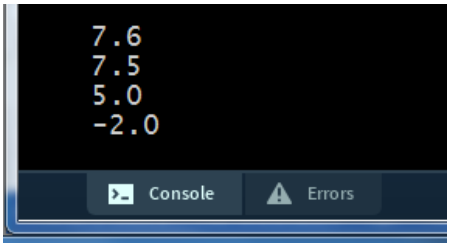
OPGAVE 5.1

Schrijf een methode die een meegegeven float (parameter) formatteert tot een String met één decimaal en deze teruggeeft.

Gebruik onderstaande code zonder (!) deze te wijzigen. Daarnaast mag je niet de ingebouwde methoden `nf()` of `format()` gebruiken, het is de bedoeling dat je de methode `formatteer(...)` echt zelf helemaal maakt.

```
void setup() {
  float getal1 = 7.56;
  float getal2 = 7.498;
  float getal3 = 4.999;
  float getal4 = -2.0001; nf()

  String tekst1 = formatteer(getal1);
  println(tekst1);

  String tekst2 = formatteer(getal2);
  println(tekst2);

  String tekst3 = formatteer(getal3);
  println(tekst3);

  String tekst4 = formatteer(getal4);
  println(tekst4);
}
```

De output is:



OPGAVE 5.2 BEHANGCASUS

Als je een muur wilt behangen, zul in één keer het juiste aantal behangrollen moeten kopen om kleurverschil te voorkomen. Dat aantal rollen kun je op de gok bepalen, maar kun je veel beter uitrekenen. In deze opdracht ga je een applicatie bouwen die dat voor je uitrekent. Deze applicaties worden in de praktijk ook gebruikt, dit is een voorbeeld: http://www.kleurmijninterieur.nl/nl/behang/behang-info/behang-calculator/2054/

EFFEN BEHANG

Effen behang is behang zonder voorgedrukte figuren e.d. Je moet er wel rekening mee houden dat je hele stroken gebruikt om op de muur te plakken, dus we willen horizontaal geen naden zien. Als je muur bijv. 2 meter hoog is en een rol behang is 11 meter lang, dan kun je daar maximaal 5 stroken uit halen (en heb je dus 1 meter afval). Maak een programma dat voor effen behang uitrekent hoeveel rollen je nodig hebt voor een muur.

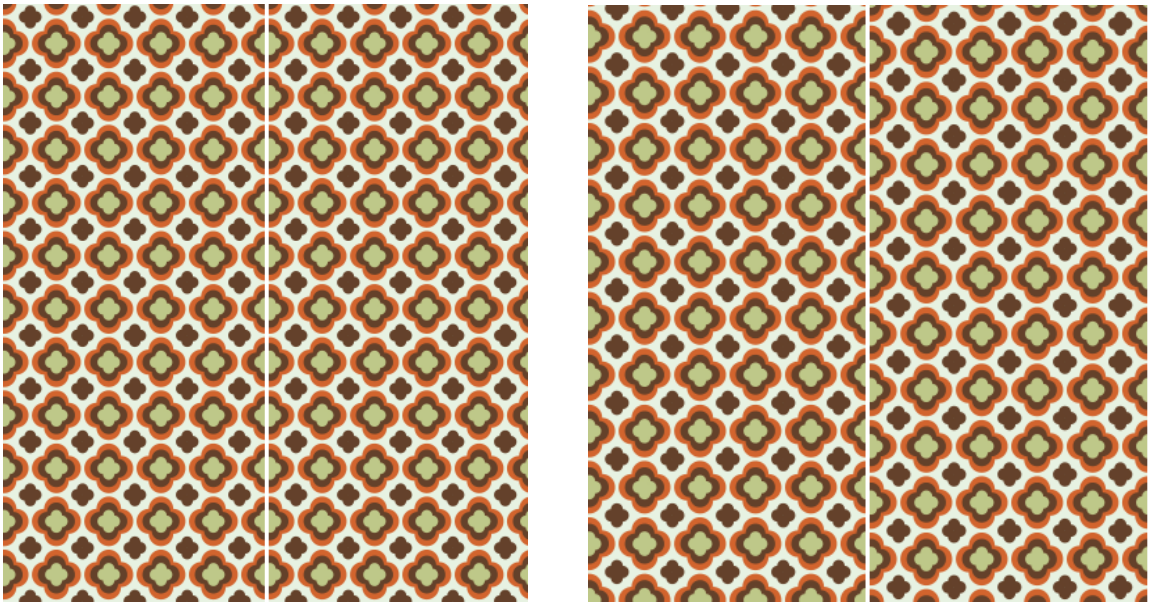
Ga uit van het volgende:

- Een standaard behangrol is 52 cm breed en 10 meter lang.
- Voor het gemak gaan we uit van een blinde muur, m.a.w. in de muur zijn geen ramen en/of deuren.
- We geven alleen de breedte en de hoogte van een muur op, bijv. de muur is 2.25m hoog en 5.50m breed.

Maak een methode waaraan je de maten van de muur en de rol meegeeft. Die methode geeft het aantal rollen dat je moet kopen terug. Begin niet gelijk met programmeren, reken eerst eens op papier uit hoeveel rollen je nodig denkt te hebben. Controleer dat antwoord eens met die van je klasgenoten.

BEHANG MET EEN PATROON

Het meeste behang is niet effen, maar is bedrukt met een patroon. Vroeger was bijvoorbeeld bloemetjesbehang erg populair. Die patronen herhalen zich steeds om een aantal centimeter, dat wordt ook wel de patroonhoogte genoemd. Als je dus gaat behangen, moet je dat behang wel ‘op patroon hangen’, m.a.w. het patroon mag niet verspringen.



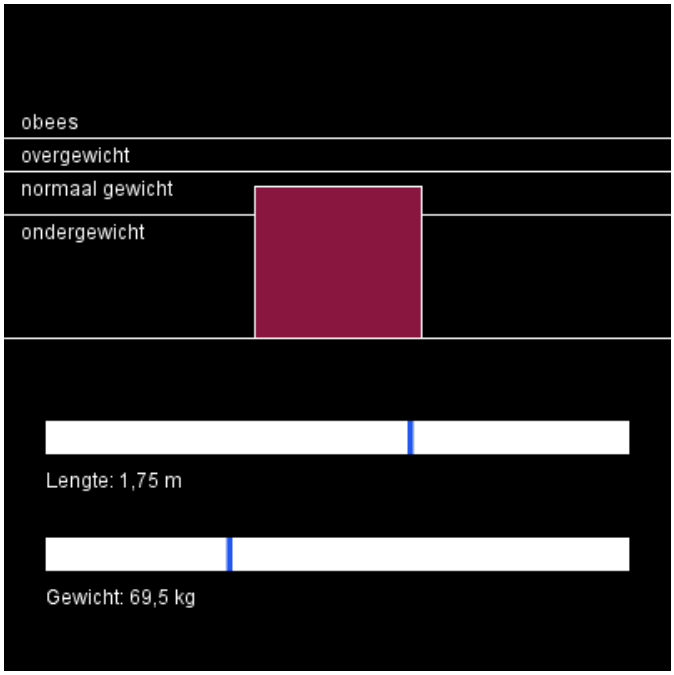
Het behang links is mooi op patroon geplakt, rechts niet (de witte naad is ter illustratie, normaal zie je die niet).

Breid de eerste oplossing uit zodat de applicatie rekening houd met patronen. Tip, de lengte per baan is dus (waarschijnlijk) groter dan de muurhoogte.

MODULEOPGAVE 5: BMI-GRAFIEK MET SLIDERS

In deze opgave ga je de uitwerking van de bmi-grafiek uit module combineren met de sliders uit de screencasts.

Maak twee sliders, één voor het invoeren van de lengten en één voor de invoer van het gewicht. Zorg ervoor dat de gebruiker de waarde van een slider kan aanpassen door de muis ingedrukt te houden (mousePressed). Zodra de gebruiker de muis loslaat (mouseRelease), blijft het sliderbalkje op zijn plek. Bereken op basis van deze twee waarden de BMI en teken een staaf waarvan de hoogte overeenkomt met de berekende BMI-waarde. Zorg er ook voor dat de verschillende categorieën zichtbaar zijn.



Figuur 1: Screenshot van een mogelijke uitwerking, je mag de sliders ook boven de grafiek plaatsen

Zorg ervoor dat de slider-code, de bmi-code en de code van het hoofdprogramma elk in zijn eigen tabblad te vinden is.

Opmerking I: het is nadrukkelijk de bedoeling dat je de code uit moduleopgave 2 (behalve de numberBox) en de code uit de screencast hergebruikt op een overzichtelijke manier.

Opmerking II: voor het bereik van de verschillende sliders kun je onderstaande waarden gebruiken.

Lengte: van 50,00 tot en met 250,00 cm

Gewicht: van 10,0 tot en met 200,0 kg

DEEL 1

Maak een analyse (met dezelfde onderdelen zoals in de vorige moduleopgaven) van alle benodigde onderdelen en beschrijf deze in een ontwerp.

DEEL 2

Bedenk wat er mis zou kunnen gaan en hoe je kan testen of het programma met deze situaties goed omgaat.

DEEL 3

Implementeer het programma volgens je ontwerp uit deel 1 en test het (gebruikmakend van de resultaten van deel 2).

EXTRA OPGAVEN OPDELEN VAN CODE IN FUNCTIES

EXTRA OPGAVE 5.1: CASUS JACK & JILL

Een bekend Engels gedichtje gaat over Jack en Jill die een heuvel (hill) op klommen om water te halen (http://en.wikipedia.org/wiki/Jack_and_Jill_%28nursery_rhyme%29). Met onderstaande code kan de Body-Mass-Index (ook wel BMI of queteletindex) van Jack, Jill en natuurlijk de heuvel berekend worden.


```
float bmi;
String naam;
float massa;
float lengte;

void setup() {
  bmiJack();
  bmiJill();
  bmiHill();
}

void bmiJack() {
  naam = "Jack";
  massa = 85;
  lengte = 1.80;
  bmi();
  printGegevens();
}

void bmiJill() {
  naam = "Jill";
  massa = 65;
  lengte = 1.65;
  bmi();
  printGegevens();
}

void bmiHill() {
  naam = "Hill";
  massa = 10000;
  lengte = 30.0;
  bmi();
  printGegevens();
}

void printGegevens() {
  println(naam + " weegt " + massa + " kilo en is " + lengte + " meter lang
(BMI=" + bmi + ")");
}

void bmi() {
  bmi = massa / (lengte*lengte);
}
```

OPDRACHT

In deze code wordt een aantal variabelen in de loop van het programma voor verschillende doeleinden gebruikt (de ene keer is “lengte” de lengte van Jack, dan van Jill en dan weer van de heuvel). Dat is niet erg fraai. Bovendien hebben de personages nu allemaal hun eigen BMI-functie die heel veel op elkaar lijken.

Herschrijf de code, zodat zo veel mogelijk gebruik gemaakt wordt van korte functies die elk een duidelijke taak hebben. Verwijder dubbele code zo veel mogelijk, en zorg ervoor dat er zo weinig mogelijk globale variabelen gebruikt worden. Let op de namen van de functies, en denk goed na over parameters en return-waarden.

EXTRA OPGAVE 5.2: CASUS ABC-FORMULE

In de wiskunde is er een bekende formule die je kan helpen de nulpunten te vinden in een kwadratische vergelijking: de ABC-formule of wortelformule. Voor deze opgave maakt het niet uit of je de formule kent, begrijpt of ermee kunt werken: zelfs als je er nog nooit van gehoord hebt, kun je de opgave maken.

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

In de onderstaande code zie je hoe de ABC-formule kan worden opgelost met behulp van een aantal hulpfuncties die elk een deel van het probleem oplossen.

```
float a;
float b;
float c;
float minb;
float kwadraatb;
float discriminant;
float wortelDiscriminant;
float noemer;
float oplossing1;
float oplossing2;

void setup() {
  a = 2;
  b = 5;
  c = -7;
  berekenMinb();
  berekenDiscriminant();
  berekenWortelDiscriminant();
  berekenNoemer();
  oplossing1 = (minb + wortelDiscriminant) / noemer;
  oplossing2 = (minb - wortelDiscriminant) / noemer;
  println(oplossing1);
  println(oplossing2);
}

void berekenDiscriminant() {
  berekenKwadraatb();
  discriminant = kwadraatb - 4*a*c;
}

void berekenWortelDiscriminant() {
  wortelDiscriminant = (float)Math.sqrt(discriminant);
  // Je hoeft niet te weten hoe deze functie precies
  // werkt, het is genoeg om te weten dat hij de
  // wortel van de discriminant in de variabele
  // wortelDiscrimant zet.
  // Je hoeft het deel "(float)Math.sqrt" dus niet
  // te begrijpen.
```

```
}

void berekenMinb() {
  minb = -b;
}

void berekenKwadraatb() {
  kwadraatb = b*b;
}

void berekenNoemer() {
  noemer = 2*a;
}
```

OPDRACHT

Herschrijf de code zo dat er geen enkele globale variabele meer wordt gebruikt, maar blijf wel evenveel functies gebruiken. Je zult de functies dus moeten herschrijven zodat ze gebruik maken van parameters en returnwaarden. Herschrijf de functies meteen zo veel mogelijk zodat ze algemeen bruikbaar worden (je ziet nu bijvoorbeeld een functie die de wortel van de discriminant kan berekenen: mooier is een functie die van elk willekeurig getal de wortel kan berekenen en het resultaat als returnwaarde geeft). Hernoem de functies wanneer nodig ook, zodat de naam de lading dekt.