SEB BP2: MODULE 1: STUKJE HERHALING, GEHEUGENMODEL, KLASSEN & OBJECTEN

We beginnen met een stukje herhaling vanuit het vorige deel (BP1). Dit als opstapje richting het object georiënteerd denken. Loop in je eigen tempo door het eerste stuk heen. Voor een aantal zal dit zeer snel gaan maar het is belangrijk dat je dit goed snapt.

STUKJE HERHALING

Onderstaand een aantal oefeningen en voorbeelden die de basiskennis nog een keer herhalen.

ONDERDEEL A

Wanneer je twee gehele getallen door elkaar deelt, krijg je niet automatisch een kommagetal, terwijl dat soms wel wenselijk is.

Bijvoorbeeld: `println(5 / 2);` levert 2, terwijl het handig kan zijn dat er 2.5 uit zou komen.

Maak een functie, `floatDelen`, die twee gehele getallen heeft als invoer. De functie deelt het eerste getal door het tweede getal en retourneert het resultaat als een kommagetal.

ONDERDEEL B

Test het programma door in het hoofdprogramma de functie een paar keer aan te roepen, het resultaat naar het scherm te schrijven en te kijken of het resultaat op het scherm overeenkomt met het resultaat dat je verwacht.

OPGAVE GLOBAAL DELEN

Gegeven onderstaande oplossing:

```
int getal1, getal2;
float resultaat;

void setup() {
  getal1 = 5;
  getal2 = 2;
  floatDelen();
  println(resultaat);
}

void floatDelen() {
  float f1 = (float)getal1;
  float f2 = (float)getal2;
  resultaat = f1 / f2;
}
```

Bovenstaande oplossing werkt wel, maar is niet ideaal. Geef een nadeel van deze uitwerking.

Tags: globale variabelen <> lokale variabelen, returnwaarden, parameters, argumenten

OPGAVE DELEN DOOR NUL

Onderstaande programma's veroorzaken drie verschillende foutenmeldingen:

A

```
void setup() {
  println(floatDelen(5, 2);
}
```

B

```
void setup() {
  println(floatDelen(5, 2.0));
}
```

C

```
void setup() {
  println(1 / 0);
}
```

Hoewel er drie verschillende fouten worden veroorzaakt, kun je deze drie fouten in twee soorten onderverdelen. Welke twee fouten horen bij dezelfde soort en welke fout is van de andere soort. Waarom kies je voor deze indeling?

Verzin een passende naam voor beide soorten foutmelding

Tags: runtime, delen door 0, syntax, compile-time, datatypes, statische types

OPGAVE ZOEKEN IN EEN ARRAY

De functie `komtGetalVoorIn(int getal, int[] lijst)` retourneert true als `getal` voorkomt in `lijst`. Als `getal` niet voorkomt in `lijst`, dan wordt er false geretourneerd.

ONDERDEEL A

Maak eerst een programma waarmee je de functie `komtGetalVoorIn` kunt testen:

1. Verzin waarden voor de twee parameters van de functie waarvan je de uitkomst van de functieaanroep uit het hoofd kunt bepalen.
2. Roep de functie aan in setup van processing (zonder de functie te definiëren).
3. Schrijf het resultaat van deze functieaanroep naar de console en controleer of je de foutmelding krijgt die aangeeft dat de functie nog niet bestaat.

ONDERDEEL B

Implementeer de functie `komtGetalVoorIn` en controleer of het testresultaat overeenkomt met wat je verwacht.

ONDERDEEL C

Wanneer is een test goed?
Tags: testen (eerst), array, testen, methodeaanroep <> methodedefinitie, return

OPGAVE DOEFUNCTIE?

Gegeven onderstaande code waarin een functie voorkomt met een onbruikbare functienaam.

```
int[][] hetVeld = {
                    {1, 6, 3},
                    {3, 2, 9},
                  };

void setup() {
  println(doeFunctie(hetVeld, 1));
}

int doeFunctie(int[][] a, int b) {
  int c = 0;
  int[] d = a[b];
  for (int i = 0; i < d.length; i++) {
    c += d[i];
  }
  return c;
}
```

ONDERDEEL A

Loop de functie, `doeFunctie`, regel voor regel door en houd bij welke variabelen er zijn en wat de waarde van elke variabele is (in feite speel je nu zelf de runtime-omgeving na). Probeer er op deze manier te achter te komen wat de functie doet.

ONDERDEEL B

Verander in de aanroep van `doeFunctie` het tweede argument in 2 (ipv 1). Welk soort foutmelding krijg je nu? Wat betekent de foutmelding?

ONDERDEEL C

Als je de variabele `a`, `b`, `c` of `d` zou aanroepen in `setup`, dan gaat dit mis. Andersom (`hetVeld` aanroepen in `doeFunctie`) gaat wel goed. Hoe komt dit? Geef in de runtime administratie die je bij opgave A hebt gemaakt aan welke variabelen bij welke functie horen.

Tags: runtime <> compile-time, stap voor stap doorlopen, runtime administratie, runtime omgeving, ArrayIndexOutOfBoundsException, Exception, locale variabele

OPGAVE DOEKEERTWEE

```
int testGetal = 5;
int[] testGetallen = {5, 5};

void setup() {
  doeKeerTwee(testGetal);
  doeKeerTwee(testGetallen);

  println(testGetal);
  println(testGetallen);
}

void doeKeerTwee(int getal) {
  getal = 2 * getal;
}

void doeKeerTwee(int[] getallen) {
  for (int i = 0; i < getallen.length; i++) {
    getallen[i] = 2 * getallen[i];
  }
}
```

Voer dit programma uit en bekijk de uitvoer. Vergelijk de waarde van het `testGetal` met de waarde in `testGetallen`. Wat is het belangrijkste verschil?

Tags: referentietypes, primitieve types, runtime administratie, overloading

OEFENINGEN

OPGAVE ARRAYS BOUWEN

OPGAVE A GETALLEN UIT TWEE ARRAYS BIJ ELKAAR OPTELLEN

Maak de methode `telElementenOp` die twee integer arrays als invoer heeft. De methode telt de getallen van elk element uit beide arrays bij elkaar op en slaat de resultaten op in een nieuwe array. Vervolgens retourneert de methode de nieuwe array.

Test deze methode in de setup-methode van processing.

Hint: ga ervan uit dat beide arrays dezelfde lengte hebben

OPGAVE B MAXIMUM BEPALEN VAN TWEE ARRAYS

Maak de methode `maakMaxArray`. Deze methode krijgt twee arrays als invoer, bepaalt de methode per element uit beide arrays het getal met de hoogste waarde en retourneert een nieuwe array met deze getallen.

Als één van beide arrays langer is dan de ander, dan moet het resultaat aangevuld worden met de elementen uit de langste array.

Test deze methode in de setup-methode van processing.

HET GEHEUGENMODEL

THEORIE

SCREENCAST ONDERWERP HET GEHEUGENMODEL

http://www.youtube.com/playlist?list=PLpd9jJvk1PjmtR_LDjx6Ao8ddS5Q_-30a

OPGAVE GEHEUGENMODEL VOLGORDE

Hieronder staan een aantal acties die in het geheugenmodel plaats kunnen vinden. Zet ze in de juiste volgorde.

a) Lokale variabelen een waarde geven.
b) Stack frame verwijderen.
c) Returnwaarde kopiëren.
d) Stack frame plaatsen.
e) Globale variabelen plaatsen.
f) Lokale variabelen plaatsen.

Tags: geheugenmodel, stack, stack frame, return,

OPGAVE PIJL IN HET GEHEUGENMODEL

ONDERDEEL A

Geef zo duidelijk mogelijk aan wat een pijl in het geheugenmodel precies betekent.

ONDERDEEL B

Geef aan waar deze pijl precies moet beginnen en waar deze pijl precies naar wijst.

Tags: geheugenmodel, variabele, array, referentie, geheugenadres

OPGAVE GEHEUGENMODEL VAN DOEKEERTWEE

ONDERDEEL A

In onderstaande code is de functie `doeKeerTwee` gegeven:

```
01  int testGetal = 5;
02
03  void setup() {
04    doeKeerTwee(testGetal);
05    println(testGetal);
06  }
07
08  void doeKeerTwee(int getal) {
09    getal = 2 * getal;
10  }
```

Teken het geheugenmodel op het moment dat `doeKeerTwee` op regel 4 is uitgevoerd, maar het stack frame van deze functie nog niet is verwijderd.

ONDERDEEL B

Hieronder staat de functie `doeKeerTwee`

```
01  int[] testGetallen = {5, 5};
02
03  void setup() {
04    doeKeerTwee(testGetallen);
05
06    println(testGetallen);
07  }
08
09  void doeKeerTwee(int[] getallen) {
10    for (int i = 0; i < getallen.length; i++) {
11      getallen[i] = 2 * getallen[i];
12    }
13  }
```

Teken het geheugenmodel op het moment dat `doeKeerTwee` op regel 4 is uitgevoerd, maar het stack frame van deze functie nog niet is verwijderd.

ONDERDEEL C

Vergelijk de geheugenmodellen uit Onderdeel A en Onderdeel B met elkaar en verklaar aan de hand van deze modellen waardoor de globale variabele `testGetal` niet van waarde is veranderd, maar de `testGetallen` wel.

Tags: stap voor stap doorlopen, geheugenmodel, referentievariabele, primitieve variabele, lokale variabele, globale variabele.

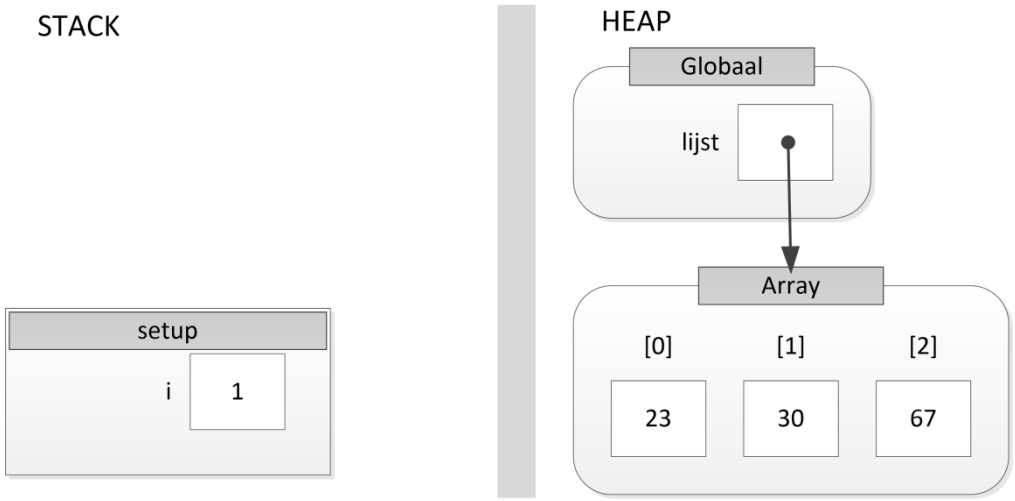
OEFENINGEN

OPGAVE LENGTH

Gegeven onderstaand programma

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

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

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

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

ONDERDEEL D

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

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

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

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?

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

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  void setup() {
02    int hetGetal = 10;
03    int[] deLijst = maakLijstMetEenGetal(2, hetGetal);
04  }
    // .. 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.

KLASSEN & OBJECTEN

THEORIE

SCREENCAST ONDERWERP KLASSEN EN OBJECTEN

http://www.youtube.com/watch?v=1sAZozVIogQ&feature=share&list=PLpd9jJvk1PjmB_VNDp61-94kAbUHqcziD

http://www.youtube.com/watch?v=GXuor-sAxFQ&feature=share&list=PLpd9jJvk1PjmB_VNDp61-94kAbUHqcziD

OPTIONEEL: PLURALSIGHT (AANRADER)

Representing complex Types with classes + Class Initializers and Constructors

https://app.pluralsight.com/player?course=java-fundamentals-language&author=jim-wilson&name=java-fundamentals-language-m5&clip=0&mode=live

OPTIONEEL: BOEK

HOOFDSTUK 1

1.1 pagina 2 en 3

1.4 t/m 1.7 (pagina 5 t/m 8 alleen de concepten bestuderen)

HOOFDSTUK2

2.3 t/m 2.4 (pagina 25 t/m 32, het sleutelwoord public kun je nu negeren)

HOOFDSTUK 3

3.12.2 (pagina 92 t/m 94)

HOOFDSTUK 4

4.14.2 (pagina 143)

OPGAVE PRODUCT MET CONSTRUCTOR

Maak een constructor voor de klasse Product (zie hiervoor, blz 11) waarmee de naam en de prijs kunt initialiseren. Gebruik deze constructor om beide producten te initialiseren in de setup-functie van Processing.

OPGAVE DAMSTEEN DEEL 1

OPGAVE A

Maak de klasse waarmee je damstenen kunt maken. Een damsteen is rond en moet een x- en y-punt hebben, een kleur en een diameter. Maak een constructor waarmee de gebruiker van de klasse alle eigenschappen van een damsteen kan meegeven.

OPGAVE B

Test de klasse damsteen door in het hoofdprogramma twee damstenen te maken: een witte en een zwarte en deze op het tekenvenster van Processing te tekenen.

OPGAVE STUDENT NULL

Gegeven onderstaande code:

```
01  void setup() {
02    Student s = new Student("kareltje", 12);
03    println(s.naam);
04  }
05
06  class Student {
07    String naam;
08    int nummer;
09
10    Student(String naam, int nummer) {
11      naam = naam;
12      nummer = nummer;
13    }
14  }
```

Wanneer deze code uitgevoerd wordt, komt er in het uitvoervenster null te staan en geen “kareltje”. Teken het geheugenmodel op regel 3 (nog voordat het constructor stack frame is verwijderd van de stack), teken daarna het geheugenmodel op regel 12.

Verklaar aan de hand van deze schetsen, hoe het komt dat er null in het uitvoervenster komt te staan.

AANVULLENDE OEFENINGEN

OPGAVE DAMSTENEN DEEL 2

ONDERDEEL A

Zie opgave op vorige bladzijde. Maak in het hoofdprogramma een array met vier damstenen: twee voor de witte speler en twee voor de zwarte speler. Initialiseer deze array en vul de array met vier nieuwe damsteenobjecten.

ONDERDEEL B

Pas de code in de draw-lus zo aan dat alle damstenen uit de array getekend worden.

Tags: array met objecten

OPGAVE PERSOONSVERANDERING

Klassikaal, begrip

ONDERDEEL A

Gegeven onderstaande code:

```
01  class Persoon {
02    String naam;
03
04    Persoon(String naam) {
05     this.naam = naam;
06    }
07  }
08
09  void setup() {
10    Persoon p1 = new Persoon("han");
11    Persoon p2 = new Persoon("ica");
12
13    p1.naam = p2.naam;
14    p2.naam = "kareltje";
15
16    println(p1.naam);
17  }
```

Teken het geheugenmodel op het moment dat `println(p1.naam)` op regel 16 wordt uitgevoerd en verklaar de uitvoer in de console (je hoeft het stack frame van `println` niet te tekenen).

ONDERDEEL B

Het programma wordt een beetje aangepast (zie vetgedrukte code) :

```
01  class Persoon {
02    String naam;
03
04    Persoon(String naam) {
05     this.naam = naam;
06    }
07  }
08
09  void setup() {
10    Persoon p1 = new Persoon("han");
11    Persoon p2 = new Persoon("ica");
12
13    p1 = p2;
14    p2.naam = "kareltje";
15
16    println(p1.naam);
17  }
```

Teken het geheugenmodel op het moment dat `println(p1.naam)` op regel 16 wordt uitgevoerd (zonder stack frame van `println`) en verklaar het verschil van de uitvoer in de console met onderdeel A.

Tags: geheugenmodel, primitieve types, referentietypes

REFLECTIEOPGAVEN

OPGAVE TESTEN

Bij verschillende opgaven is er gevraagd om een test te schrijven voor een functie.

Op welke plek, of plekken schrijf je in een processingprogramma die test?

Geef een schets van de code die je maakt, als je een test moet schrijven.

Wanneer is de test van goede kwaliteit?

OPGAVE FOUTMELDINGEN

Er zijn een aantal belangrijke foutmeldingen besproken.

Welke foutmeldingen zijn dit? Waardoor wordt elke foutmelding veroorzaakt? Geef per foutmelding de oplossing.

OPGAVE PROGRAMMEERSTIJL

Het hanteren van een goede programmeerstijl is een paar keer teruggekomen. Zoek de opgaven op waarin programmeerstijl aan de orde is geweest. Geef een paar vuistregels voor een goede programmeerstijl.

OPGAVEN BEGRIPPEN VERBINDEN

Hieronder staan verschillende groepen met begrippen. Maak een zin waarin je elk begrip verbindt door de constructie: “bevat/kan bevatten”, “is/kan zijn”, “wordt weergegeven door/kan weergegeven worden door”, “zorgt voor/kan zorgen voor.

Je mag de begrippen meervoud maken als dit de zin ten goede komt.

Bijvoorbeeld:

Rechthoek, Stack frame, Geheugenmodel

Levert

Het geheugenmodel kan stack frames bevatten die elk worden weergegeven door een rechthoek.

- Runtime administratie, geheugenmodel, variabelen.
- Stack frame, geheugenmodel, functieaanroep.
- Primitieve variabele, referentievariabele, object, array, integer
- Geheugenadres, geheugenmodel, pijl.
- Lokale variabele, parameter
- Globale variabele declaratie, lokale variabele declaratie, functiedefinitie
- Functiedefinitie, functieaanroep.

OPGAVE OPGAVEN EN TAGS

ONDERDEEL A

Bekijk nogmaals elke opgave en bekijk welke opgaven je hebt gemaakt, welke opgaven je denkt te beheersen en van welke opgave je onderdelen nog niet begrijpt.

ONDERDEEL B

Bij elke opgave staan tags genoemd. Benoem de rol van elke tag in de opgave, of vraag dit na. Ga ook na of er een tag bij de opgave mist.