SEB BP2: MODULE 2: METHODEN, INTERACTIE, NAAR JAVA

Deze week gaan we in rap tempo verder. We gaan eerst kijken naar methoden binnen classes. Vervolgens willen we ook gaan kijken hoe classes onderling met elkaar kunnen communiceren. We zullen ook overstappen op Eclipse als ontwikkelomgeving voor Java. Voor velen een openbaring en voor anderen ook weer even wennen.

THEORIE METHODEN

SCREENCAST ONDERWERP: METHODEN

http://www.youtube.com/watch?v=g8xMStW4lT0&feature=share&list=PLpd9jJvk1PjmB_VNDp61-94kAbUHqcziD

OPTIONEEL: PLURALSIGHT (ZELFDE ALS MODULE 1)

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

HOOFDSTUK2

2.5 t/m 2.7 (pagina 32 t/m 37)

2.16 en 2.17 (pagina 51 t/m 54)

OPGAVE DAMSTEEN

ONDERDEEL A

In module 1 heb je een functie gemaakt voor het tekenen van een damsteen. Maak van deze een functie een methode in de klasse Damsteen waarin je alle code plaatst om één damsteen te tekenen.

ONDERDEEL B

Test de methode `tekenDamsteen` voor alle damstenen uit de array in de draw-lus van processing.

ONDERDEEL C

Voeg de eigenschap `isGeselecteerd` die de waarde true, of false kan hebben toe aan de klasse Damsteen. Zorg ervoor dat de methode `tekenDamsteen` een dikke rand (met de processing-methoden `stroke`, `strokeWeight` en `noStroke`) om de damsteen heen tekent als `isGeselecteerd` de waarde true heeft. Je hoeft niet zelf te detecteren of de damsteen geselecteerd is: het wordt gewoon een waarde die van buiten aan- of uitgezet kan worden.

ONDERDEEL D

Test de methode `isGeselecteerd`. Doe dit door in de setup van processing de eigenschap `isGeselecteerd` van één van de damstenen op true te zetten. Zet dus in je code de boolean `isGeselecteerd` op true: je hoeft niet te detecteren of de muis in de buurt van de steen is of iets dergelijks.

OPGAVE SLIDER

Plak onderstaande code in een nieuw venster van Processing.

```
float s1X, s1Y, s1Breedte, s1Hoogte;
int s1NPosities;

void setup() {
  size(300, 200);
  background(0);

  s1Breedte = 200;
  s1Hoogte = 50;
  s1X = (width - s1Breedte) / 2;
  s1Y = 50;
  s1NPosities = 5;
}
```

In deze opgave ga je deze code verplaatsen naar een klassendefinitie maken van een Slider.

ONDERDEEL A

De variabelen `s1X`, `s1Y`, `s1Breedte`, `s1Hoogte` en `s1NPosities` zijn eigenschappen van een slider. Maak een klasse Slider en plaats deze variabelen op de goede manier binnen deze klasse. Pas daarbij ook de namen op de goede manier aan.

ONDERDEEL B

De meeste code in de setup is eigenlijk initialisatiecode van de slider. Deze code kan beter in een constructor. Voeg een constructor toe aan de klasse `Slider` en test deze constructor in de setup van Processing

OPGAVE SLIDER DEEL 2

Hieronder is de draw lus van de slider gegeven. Verander de functies uit de oude code naar methoden van de klasse slider en test de nieuwe klasse slider in de draw-lus van processing.

```
void draw() {
  int s1HuidigePositie = bepaalSliderPositie(s1X, s1Breedte,
s1NPosities);
  tekenSlider(s1X, s1Y, s1Breedte, s1Hoogte, s1HuidigePositie,
s1NPosities);
}
void tekenSlider(float x, float y, float breedte, float hoogte,
                 int positie, int nPosities) {
```

```
  float blokjeBreedte = breedte / nPosities;

  noStroke();
  fill(255);
  rect(x, y, breedte, hoogte);

  fill(#2257F0);
  rect(x + positie * blokjeBreedte, y, blokjeBreedte, hoogte);
}

int bepaalSliderPositie(float x, float breedte, int nPosities) {
  float blokjeBreedte = breedte / nPosities;

  if (mouseX < x) {
   return 0;
  }
  else if (mouseX >= breedte + x) {
    return nPosities - 1;
  }
  else {
    return floor((mouseX  - x) / blokjeBreedte);
  }
}
```

OPGAVE TOSTRING()

PERSOON

Voeg aan de klasse Persoon de methode `toString` die een relevante string retourneert. Test de methoden in het hoofdprogramma.

DAMSTEEN

Voeg aan de klasse Damsteen de methode `toString` die een relevante string retourneert. Test de methoden in het hoofdprogramma.

OPGAVE KAARTAUTOMAAT

De klasse KaartjesAutomaat simuleert een kaartautomaat bij een bioscoop die kaartjes kan leveren voor één film met één prijs. De film en de prijs van een kaartje wordt ingesteld via de constructor. Instanties zorgen ervoor dat een gebruiker alleen bedragen inwerpen die groter zijn dan 0 en dat de automaat alleen een kaartje afdrukt als er voldoende geld 'ingeworpen' is. Daarnaast kan het wisselgeld bepaald en teruggegeven worden. Ook wordt het totaal ingeworpen bedrag bijgehouden en kan de automaat geleegd worden (waardoor het totaal weer op 0 komt te staan). De afdruk van de kaartjes wordt gesimuleerd door de naam van het kaartje en de prijs naar de console te schrijven (eventueel samen met artistieke ascii-art).

ONDERDEEL A MAAK HET KLASSENONTWERP

Teken het klassendiagram met de naam van de klasse, de eigenschappen van de velden (inclusief type) en de methoden (inclusief parameters met type en returntype).

OPGAVE B IMPLEMENTEREN EN TESTEN

IMPLEMENTATIETEAM

Maak deze klasse en zorg ervoor dat de code zo goed mogelijk omgaat met foutieve invoer.

TESTTEAM

Maak het hoofdprogramma waarin je één instantie van deze klasse test. Test door alle uitvoer met println naar het uitvoerscherm te schrijven. Verzin invoer en bedenk welke uitvoer je wilt hebben. Probeer zo grondig mogelijk te testen door naast geldige invoer ook foutieve invoer aan de ticketmachine te voeren.

OEFENINGEN

OPGAVE MEER KLASSENDEFINITIES

ONDERDEEL A STUDENT

Maak de klasse student. Een student heeft een naam (voornaam en achternaam) en een nummer. Maak een bijpassende constructor en test de klasse in het hoofdprogramma

ONDERDEEL B DOBBELSTEEN

Maak de klasse Dobbelsteen. Verzin zelf welke eigenschap(pen) een dobbelsteen moet hebben. Maak een bijpassende constructor en test de klasse in het hoofdprogramma.

OPGAVE SLIDER TOSTRING()

Voeg aan de klasse Slider de methode toString die een relevante string retourneert. Test de methoden in het hoofdprogramma.

OPGAVE BESTAANDE CONSTRUCTOR GEBRUIKEN

Maak een tweede constructor voor klok waarmee je ook de tijd kunt initialiseren. Voorkom herhalende code door in de nieuwe constructor gebruik te maken van de constructor die al bestaat en de methode setTijd.

Hieronder is alvast de definitie:

```
Klok(float x, float y, float breedte, int uren, int minuten) {
}
```

UITDAGING: WAT IS EEN STRING (& HOE VERHOUDT ZICH DAT TOT OBJECTEN)

Draai onderstaande code en bekijk de uitvoer in de console.

```
class Persoon {
  String naam;

  Persoon(String naam) {
    this.naam = naam;
  }
}

void setup() {
  String t = "hallo";
  verander(t);
  println(t);

  int g = 10;
  verander(g);
```

```
  println(g);

  Persoon p = new Persoon("han");
  verander(p);
  println(p.naam);
  veranderAnders(p);
  println(p.naam);
}

void verander(String tekst) {
  tekst = "veranderd";
}

void verander(int getal) {
  getal = 0;
}

void verander(Persoon persoon) {
  persoon.naam = "kareltje";
}

void veranderAnders(Persoon persoon) {
  persoon = new Persoon("pietje puk");
}
```

OPGAVE A

Is een String op basis van de uitvoer een referentietype, of een primitief type.

OPGAVE B

Hoewel een String zich niet lijkt te gedragen als een referentietype, blijkt een String wel een referentietype te zijn. Zoek uit wat er aan de hand is met Strings en teken het geheugenmodel van de aanroep van `verander(t)`.

OPGAVE C

Leg aan de hand van het geheugenmodel uit waardoor `println(t)` nog steeds de waarde `t` afdrukt.

THEORIE INTERACTIE

SCREENCAST ONDERWERP INTERACTIE TUSSEN OBJECTEN

http://www.youtube.com/playlist?list=PLpd9jJvk1Pjlt8c30X9uu7N9r2aQVey93

OPTIONEEL: PLURALSIGHT (ZELFDE ALS MODULE 1)

A Closer Look at Parameters

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

OPTIONEEL: BOEK

Hoofdstuk 3

3.1 t/m 3.11 (pagina 69 t/m 90)

3.15 (pagina 99 t/m 101)

OPGAVE DAMBORD MET STENEN

ONDERDEEL A

Maak een dambord volgens onderstaande specificatie.

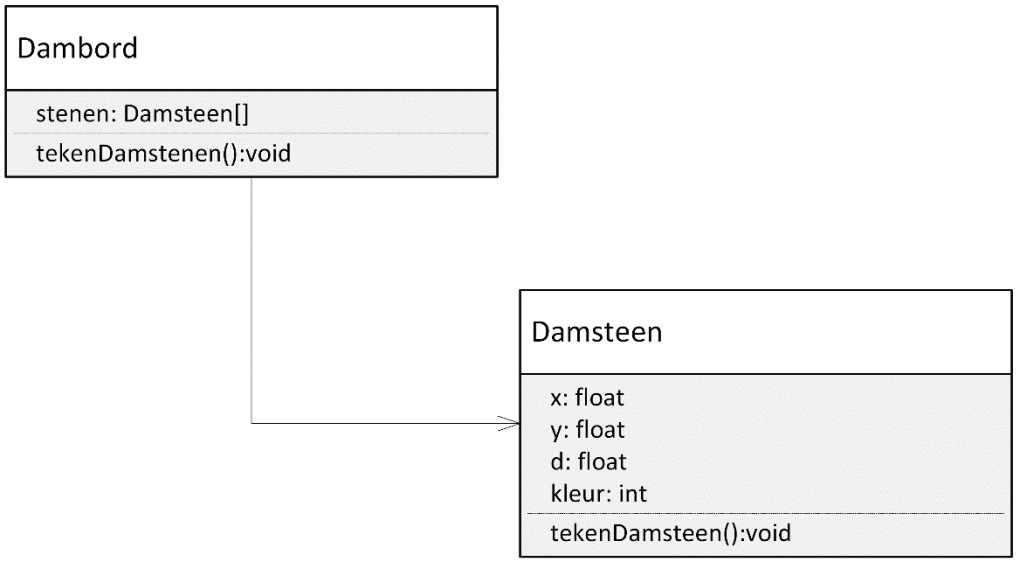
INSTANTIEVARIABELE

Een dambord houdt een Array: stenen waarin vier damstenen zitten. Twee van deze stenen zijn zwart, twee wit. De positie en de diameter van de damstenen maakt niet uit als de diameter van elke steen maar gelijk is.

METHODE

Een dambord heeft één methode: `tekenDamstenen` die alle damstenen die in de variabele `stenen` zit tekent.

Je kunt de volgende klassendiagram gebruiken.

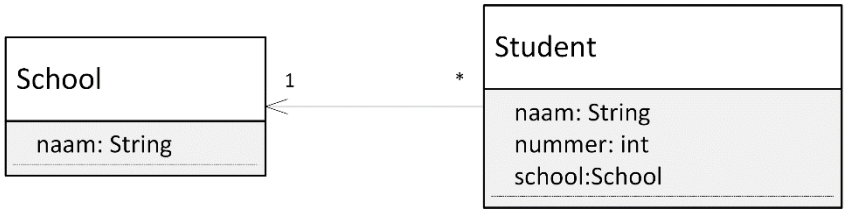


ONDERDEEL B

Test de klasse `Dambord` met bijbehorende methode `tekenDamstenen` in het hoofdprogramma.

OPGAVE STUDENT OP SCHOOL

Gegeven onderstaande klassendiagram.



Met bijbehorende code:

KLASSE SCHOOL

```
01  class School {
02    String naam;
03
04    School(String naam) {
05      this.naam = naam;
06    }
07  }
```

KLASSE STUDENT

```
01  class Student {
02    String naam;
03    int nummer;
04    School school;
```

```
05
06    Student(String naam, int nummer, String school) {
07      this.naam  = naam;
08      this.nummer = nummer;
09      this.school = new School(school);
10    }
11  }
```

HOOFDPROGRAMMA

```
01  void setup() {
02    School deSchool = new School("ICA");
03
04    Student s1 = new Student("hanneke", 1, "ICA");
05    Student s2 = new Student("femke", 2, "ICA");
06  }
```

De constructor van de klasse `Student` bevat een fout die vaak voorkomt en makkelijk verborgen blijft. Het datatype van de parameter `school` is onhandig, waardoor in regel 9 de fout ontstaat.

ONDERDEEL A

Teken het geheugenmodel van het hoofdprogramma op het moment dat deze regel 5 net heeft uitgevoerd.

ONDERDEEL B

Leg op basis van deze schets uit wat er fout gaat.

ONDERDEEL C

Lost onderstaande verbetering op regel 4 en 5 aan het hoofdprogramma de fout op?

```
01  void setup() {
02    School deSchool = new School("ICA");
03
04    Student s1 = new Student("hanneke", 1, deSchool.naam);
05    Student s2 = new Student("femke", 2, deSchool.naam);
06  }
```

OPGAVE GEHEELGETAL

In onderstaand programma is er een klasse gemaakt voor een geheel getal. De klasse heeft één eigenschap met de waarde van het getal. Hoewel het misschien raar is om een klasse te maken voor een ding dat al bestaat, gaat het in deze opgave om het verschil tussen een klasse en primitieve te onderzoeken.

```
01  class GeheelGetal {
02    int waarde;
03
04    GeheelGetal(int startwaarde) {
05     waarde = startwaarde;
06    }
07  }
08
09  void setup() {
10    int getal1 = 10;
11    GeheelGetal getal2 = new GeheelGetal(10);
12
13    println("getal1: " + getal1);
14    println("getal2: " + getal2.waarde);
15    println("-----------------------");
16
17    telErTienBijOp(getal1);
18    telErTienBijOp(getal2);
19
20    println("getal1: " + getal1);
21    println("getal2: " + getal2.waarde);
22
23  }
24
25  void telErTienBijOp(int getal) {
26    getal += 10;
27  }
28
29  void telErTienBijOp(GeheelGetal getal) {
30    getal.waarde += 10;
31  }
```

Draai het programma in processing en bekijk de uitvoer: regels 20 en 21 leveren niet hetzelfde getal.

ONDERDEEL A

Teken het geheugenmodel op het moment dat `telErTienBijOp(getal1)` op regel 17 bezig is en hierbinnen bij regel 27 is.

ONDERDEEL B

Teken het geheugenmodel op het moment dat `telErTienBijOp(getal2)` op regel 18 bezig is en hierbinnen bij regel 31 is.

ONDERDEEL C

Verklaar op basis van de twee geheugenmodellen het verschil in de uitvoer.

ONDERDEEL D

Op regels 14 en 21 kun je niet `println(getal2)` typen, maar moet je `println(getal2.waarde)` gebruiken. Met een kleine toevoeging aan de klasse GeheelGetal is dit wel mogelijk. Welke? Waarom zouden we dit willen?

OPGAVE KAARTJESAUTOMAAT 2: MEERDERE SOORTEN KAARTJES

FUNCTIONELE INFORMATIE

De klasse KaartjesAutomaat moet nu kaartjes kunnen bijhouden van verschillende soorten films. De gebruiker van de automaat kan dan één kaartje selecteren uit het aanbod van de automaat. Natuurlijk moet het gekozen kaartje ook weer “ontkozen” kunnen worden. De overige functionaliteit is hetzelfde.

TECHNISCHE INFORMATIE

Alle informatie en al het gedrag van een kaartje moet uit de klasse KaartjesAutomaat gehaald worden en in een nieuwe klasse met de naam Kaartje worden gestopt. De klasse KaartjesAutomaat moet een array van Kaartje bijhouden. Daarnaast moet er in een instantievariabele bijgehouden worden welk Kaartje is geselecteerd. Het selecteren van het kaartje gaat door middel van de index in de array van kaartsoorten (dus niet gaan zoeken op kaartnaam o.i.d.). De lijst met kaartsoorten moet in de constructor van KaartjesAutomaat meegegeven worden en kan daarna niet worden gewijzigd.

OPGAVE A ONTWERPEN VAN DE KLASSEN

Teken het klassendiagram met de namen van de klassen, de eigenschappen van de velden (inclusief type) en de methoden (inclusief parameters met type en returntype).

Teken het geheugenmodel voor een KaartjesAutomaat met 3 kaartsoorten. Ga ervan uit dat de constructor van de KaartjesAutomaat net is uitgevoerd.

OPGAVE B ONTWERPEN VAN DE TESTEN

Ontwerp* een programma waarmee je elke methode van de nieuwe klassen kunt testen. Let op dat je per methode bedenkt of er randvoorwaarden** zijn om de methode met succes te kunnen uitvoeren. Bedenk welke foutmelding er geprint moet worden als er niet aan de randvoorwaarde is voldaan.

Schrijf de testen die je ontworpen hebt, uit in een processing programma.

*ONTWERP

In dit geval wordt met ontwerp bedoeld dat je een programma maakt waarin je elke methode van de KaartjesAutomaat en Kaartje test. Je moet dus verzinnen in welke volgorde je deze methodes aanroept en hoe vaak je dit doet en wat je met die volgorde wilt bereiken.

**RANDVOORWAARDEN

Een randvoorwaarde van de methode printKaartje bijvoorbeeld is dat het saldo groter of gelijk aan de prijs van het kaartje dat geprint moet worden. Als er niet aan deze randvoorwaarde is voldaan, kan deze methode de taak waar hij voor bedoeld is niet uitvoeren (en daar moet je als programmeur iets mee doen).

OPGAVE C TESTEN EN IMPLEMENTEREN

Los vervolgens error voor error op, door de klasse Kaartje te maken en de klasse KaartjesAutomaat aan te passen.

UITDAGING KAARTJESAUTOMAAT 3

FUNCTIONELE INFORMATIE

De KaartjesAutomaat wordt verder aangepast. Er kunnen meerdere kaartjes tegelijkertijd worden gekocht (in één bestelling). Dit kunnen verschillende soorten kaartjes zijn, maar ook dezelfde.

TECHNISCHE INFORMATIE

De kaartjes die de gebruiker gaat kopen, moeten in een nieuwe klasse met de naam Bestelling komen. Bedenk zelf hoe de bestelling de gekozen kaartjes en de hoeveelheid kaartjes van één soort gaat bijhouden. Bedenk ook zelf welke methode in welke klasse ervoor zorgt dat er meerdere kaartjes van één kaartsoort geselecteerd kunnen worden.

OPGAVE A ONTWERPEN VAN DE KLASSEN

Teken het klassendiagram met de namen van de klassen, de eigenschappen van de velden (inclusief type) en de methoden (inclusief parameters met type en returntype).

Teken het geheugenmodel voor een KaartjesAutomaat met 3 kaartsoorten in één Bestelling.

OPGAVE B ONTWERPEN VAN DE TESTEN

Ontwerp een programma waarmee je elke methode van de nieuwe klassen kunt testen. Let op dat je per methode bedenkt of er randvoorwaarden zijn om de methode met succes te kunnen uitvoeren. Print een foutmelding als er niet aan de randvoorwaarde is voldaan.

OPGAVE C TESTEN EN IMPLEMENTEREN

Schrijf de testen die je ontworpen hebt in opgave B uit in een processingprogramma.

OPGAVE D

Los vervolgens error voor error op, door de nieuwe klasse te maken en de bestaande klassen aan te passen waar nodig.

OEFENINGEN

OPGAVE VERHOUDING BREEDTE/HOOGTE IN TELLER

OPGAVE A

De verhouding tussen de breedte en de hoogte wordt nu afgevangen in klok door het statement

```
hoogte = breedte * 0.4f;
```

Zorg ervoor dat de juiste verhouding tussen de breedte en de hoogte in de klasse Teller wordt afgevangen in plaats van in de klasse Klok. Test de wijzigingen goed.

OPGAVE B

Welke implementatie geniet volgens jou de voorkeur:

A. Verhouding tussen de hoogte en de breedte in Klok
B. Verhouding tussen de hoogte en breedte in Teller
C. Maakt niet uit

Probeer zo goed mogelijk te formuleren waarom je voor jouw keuze hebt gekozen. Als je geen reden kunt verzinnen, mag je ook aangeven dat het om een vaag gevoel, of intuïtie gaat. Probeer dan dit zo helder mogelijk te beschrijven.

OPGAVE TOSTRING IN KLOK

OPGAVE A

Welke informatie zou je in de string representatie van Klok willen stoppen?

UITDAGING OPGAVE B

Implementeer de toString in Klok op basis van het antwoord op opgave A.

THEORIE: NAAR JAVA I

SCREENCAST: VAN PROCESSING NAAR JAVA

http://www.youtube.com/watch?v=Y2w9OvLTg8A&feature=share&list=PLpd9jJvk1PjnMmrtlNeOzviLhJolx0_oi

Video's 2, 3 en 4 komen VERDEROP aan bod.

OPTIONEEL: PLURALSIGHT (AANRADER)

Eclipse Guided Tour – Part 1, Hoofdstukken t/m Editing Code.

https://app.pluralsight.com/player?course=eclipse-guided-tour-part1&author=tod-gentille&name=eclipse-guided-tour-part1-m1&clip=0&mode=live

OPGAVE INSTALLEREN VAN ECLIPSE

Vanaf de komende les gaan we gebruik maken van een geavanceerdere programmeeromgeving: Eclipse. Download en installeer ter voorbereiding van deze les Eclipse alvast.

1. Ga naar www.eclipse.org/downloads
2. Download de Eclipseversie "Eclipse IDE for Java Developers"
3. Unzip het gedownloade bestand op een locatie die je gemakkelijk kunt terugvinden
4. Start "Eclipse" in de uitgepakte map
5. Kies voor de standaard workspace

OPGAVE STUDENT IN JAVA

Zorg ervoor dat de Student en de School uit de opgave StudentOpSchool (bladzijde 8) in Java draait.

OPGAVEN GEHEUGENMODEL VAN TEKENDAMSTEEN

ONDERDEEL A

Gegeven onderstaande code om een damsteen te maken.

```
class Damsteen {
  float x, y, d;
  int kleur;
  boolean isGeselecteerd;

  Damsteen(float x, float y, float d, int kleur) {
    this.x = x;
    this.y = y;
    this.d = d;
    this.kleur = kleur;
    isGeselecteerd = false;
  }

  void tekenDamsteen() {
    ellipseMode(CORNER);
```

```
    fill(kleur);
    if (isGeselecteerd) {
      stroke(255, 0, 0);
    }
    else {
      noStroke();
    }
    ellipse(x, y, d, d);
  }
}

Damsteen d1;

void setup() {
  d1 = new Damsteen(0.0, 0.0, 20.0, 255);
}

void draw() {
  d1.tekenDamsteen();
}
```

Teken in het meegeleverde diagram het stack frame van `tekenDamsteen` met de juiste lokale variabele(n). op het moment dat deze methode wordt aangeroepen op de een na laatste regel

ONDERDEEL B

De methode tekenDamsteen wordt nu uit de klassedefinitie van Damsteen gehaald en opnieuw gedefinieerd als globale functie zoals hieronder te zien is:

```
class Damsteen {
  float x, y, d;
  int kleur;
  boolean isGeselecteerd;

  Damsteen(float x, float y, float d, int kleur) {
    this.x = x;
    this.y = y;
    this.d = d;
    this.kleur = kleur;
    isGeselecteerd = false;
  }
}

void tekenDamsteen(Damsteen steen) {
  ellipseMode(CORNER);
  fill(steen.kleur);
  if (steen.isGeselecteerd) {
    stroke(255, 0, 0);
  }
```

```
  else {
    noStroke();
  }
  ellipse(steen.x, steen.y, steen.d, steen.d);
}

Damsteen d1;

void setup() {
  d1 = new Damsteen(0.0, 0.0, 20.0, 255);
}

void draw() {
  tekenDamsteen(d1);
}
```

Teken opnieuw in het meegeleverde diagram het stack frame van `tekenDamsteen` met de juiste lokale variabele(n). Op het moment dat deze methode wordt aangeroepen op de een na laatste regel

ONDERDEEL C

Versie A is beter dan versie B. Leg uit waarom.

REFLECTIEOPGAVEN

OPGAVE TESTEN VAN EEN KLASSE

Bij sommige opgaven is er gevraagd om een test te schrijven voor een klasse. Geef een schets van de code die je maakt, als je een test voor een klasse schrijft.

OPGAVE FUNCTIE <> METHODE

Wat is de overeenkomst tussen een functie en een methode? Wat is het verschil tussen een functie en een methode?

OPGAVE KLASSENDIAGRAMMEN <> GEHEUGENMODEL

Hoewel klassendiagrammen en geheugenmodellen beide informatie geven over het programma, is het soort informatie verschillend.

Leg uit voor beide diagrammen uit welke informatie eruit te halen is en op welk moment in het ontstaan van het programma je het diagram kunt gebruiken. Gebruik daarbij zoveel mogelijk van onderstaande begrippen.

THEORIE: NAAR JAVA II

SCREENCAST ONDERWERP 4.2 – 4.4 VAN PROCESSING NAAR JAVA

http://www.youtube.com/watch?v=Ub-0Pa5B1Lw&feature=share&list=PLpd9jJvk1PjnMmrtlNeOzviLhJolx0_oi

http://www.youtube.com/watch?v=guHGz_a1Z0w&feature=share&list=PLpd9jJvk1PjnMmrtlNeOzviLhJolx0_oi

http://www.youtube.com/watch?v=XQltRh1ba0g&feature=share&list=PLpd9jJvk1PjnMmrtlNeOzviLhJolx0_oi

OPTIONEEL: PLURALSIGHT (OPTIONEEL)

Eclipse Guided Tour – Part 1, Hoofdstukken t/m Advanced Navigation and Searching

https://app.pluralsight.com/player?course=eclipse-guided-tour-part1&author=tod-gentille&name=eclipse-guided-tour-part1-m1&clip=0&mode=live

BOEK (OPTIONEEL)

HOOFDSTUK 2

2.8 (pagina 37 t/m 40)

HOOFDSTUK 5

5.11 (pagina 200 t/m 203)

PROCESSING DOCUMENTATIE

Processing klassen in Eclipse gebruiken:

https://processing.org/tutorials/eclipse/

OPGAVE KLOK IN ECLIPSE

OPGAVE A

Zet alle klassen uit de KlokApp in Processing om naar Javacode in Eclipse.

OPGAVE B

Zorg ervoor dat de klasse Klok geen instantievariabelen x, y, hoogte en breedte meer heeft. Maak in klok wel getters en setters voor x, y, hoogte en breedte en zorg ervoor dat in die getters en setters de urenTeller en minutenTeller gebruikt worden om de juiste waarde terug te geven en aan te passen.

OPDRACHT KLASSEDIAGRAM CHUCK-A-LUCK

SPELBESCHRIJVING

Beschouw de volgende eenvoudige versie van het spel Chuck-a-luck: Per ronde zet je een bedrag in op een geluksgetal van 1 tot 6 en vervolgens gooi je drie dobbelstenen met behulp van een dobbelbeker. Als geen van

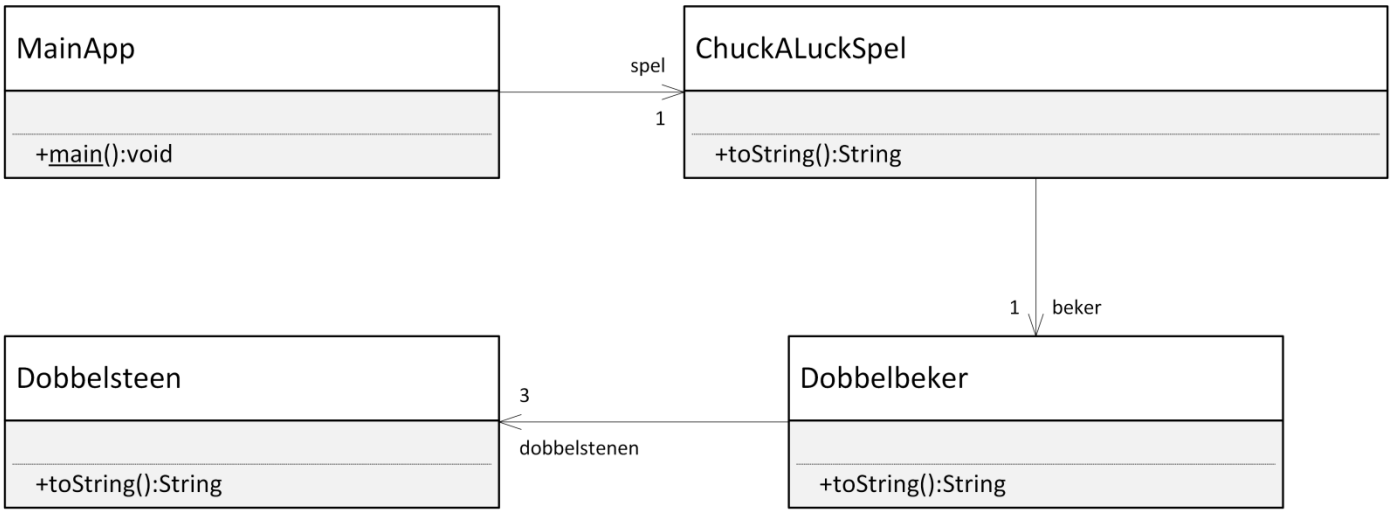
de dobbelstenen dit geluksgetal aangeeft, dan ben je je inzet kwijt. In alle andere gevallen wordt uitbetaald afhankelijk van het aantal dobbelstenen dat het voorspelde aantal ogen weergeeft:

Overeenkomende dobbelstenen	Uitbetaling
1 (een Single)	1:1
2 (een Double)	2:1
3 (een Triple)	10:1

TECHNISCHE SPECIFICATIE

KLASSEN

De hele applicatie bestaat uit vier klassen: Dobbelsteen, Dobbelbeker en ChuckALuckSpel en MainApp. Zie onderstaande klassendiagram (meeste methoden en instantievariabelen zijn weggelaten).



MAINAPP

De MainApp fungeert als een test voor de andere drie klassen. Hieronder is een mogelijke implementatie gegeven van de MainApp.

```
public class MainApp {
     public static void main(String[] args) {
          ChuckALuckSpel spel = new ChuckALuckSpel(100);

          spel.speelRonde(3, 5);
          System.out.println(spel);

          spel.speelRonde(3, 20);
          System.out.println(spel);

          spel.speelRonde(4, 10);
```

```
            System.out.println(spel);
        }
}
```

Deze code levert onderstaande uitvoer in de console:

```
-------------------------
Ronde: 1
geluksgetal: 3
worp: 1 6 1
saldo: 95
-------------------------
Ronde: 2
geluksgetal: 3
worp: 2 6 5
saldo: 75
-------------------------
Ronde: 3
geluksgetal: 4
worp: 5 3 4
saldo: 85
```

OPGAVE

Maak het klassendiagram af door in de drie klassen: ChuckALuckSpel, Dobbelbeker en Dobbelsteen alle ontbrekende instantievariabelen en methoden toe te voegen zodat ChuckALuck gespeeld kan worden op basis van bovenstaande beschrijving.

Probeer geen getters en setters te declareren tenzij je denkt dat je echt niet zonder kan in deze situatie.

OPDRACHT CHUCK-A-LUCK IMPLEMENTEREN

Implementeer chuck-a-luck op basis van de omschrijving uit de voorbereiding, het klassendiagram en alle klassen met stubs van de methoden.

WERKWIJZE

STAP 1

Probeer met 3 andere studenten af te spreken samen deze oefening te doen. Je kunt het ook alleen doen als je wilt maar dat wordt wat lastiger en moet je de discipline hebben om de test + klasse apart uit te werken. Een ieder is verantwoordelijk voor het maken van de implementatie en de tests van één klasse uit het chuck-a-luck spel. De andere studenten mogen nog niet gebruik maken van de code van andere tweetallen.

STAP 2

Schuif alle klassen pas bij elkaar nadat elke student de implementatie van de klasse af heeft.

STAP 3: EVALUEER

Ging het bij elkaar voegen van de klassen probleemloos. Zo niet, waardoor werden de problemen veroorzaakt?

OEFENINGEN

OPGAVE VAN INSTANTIEVARIABELE NAAR PARAMETER

Hieronder is een gedeelte van de klasse Teller te zien zoals deze in de screencast is gemaakt.

```
public class Teller {
     private int maximum;
     private int waarde;
     private float x, y, breedte, hoogte;
     private KlokApp app;

     public Teller(KlokApp app, int maximum, float x, float y, float
breedte) {
          this.maximum = maximum;
          waarde = 0;
          this.x = x;
          this.y = y;
          this.breedte = breedte;
          this.hoogte = breedte * 0.8f;
          this.app = app;
     }
...
...
     public void tekenTeller() {
          app.noStroke();
          app.fill(0);

          app.rectMode(app.CORNER);
          app.rect(x, y, breedte, hoogte);

          app.fill(255, 0, 0);
          app.textSize(hoogte);

          app.textAlign(app.LEFT);
          float tijdBreedte = app.textWidth(getTijdNotatie());
          float verschuivingX = (getBreedte() - tijdBreedte) / 2;
```

```
        float verschuivingY = app.textAscent() - app.textDescent() / 2;

        app.text(getTijdNotatie(), x + verschuivingX, y + verschuivingY);
    }
}
```

Als we de instantievariabele app veranderen in een parameter die we aan tekenTeller meegeven, dan krijgen we onderstaande code:

```
public class Teller {
    private int maximum;
    private int waarde;
    private float x, y, breedte, hoogte;

    public Teller(int maximum, float x, float y, float breedte) {
        this.maximum = maximum;
        waarde = 0;
        this.x = x;
        this.y = y;
        this.breedte = breedte;
        this.hoogte = breedte * 0.8f;
    }
...
...
    public void tekenTeller(KlokApp app) {
        app.noStroke();
        app.fill(0);

        app.rectMode(app.CORNER);
        app.rect(x, y, breedte, hoogte);

        app.fill(255, 0, 0);
        app.textSize(hoogte);

        app.textAlign(app.LEFT);
        float tijdBreedte = app.textWidth(getTijdNotatie());
        float verschuivingX = (getBreedte() - tijdBreedte) / 2;
        float verschuivingY = app.textAscent() - app.textDescent() / 2;

        app.text(getTijdNotatie(), x + verschuivingX, y + verschuivingY);
    }
```

```
}
```

OPGAVE A

Voer deze wijziging door in de klasse Teller en pas de code in Klok en KlokApp zo aan dat de code weer werkt.

OPGAVE B

Leg uit waar de voorkeur naar uit gaat: app als instantievariabele of een parameter.