# SEB BP2: MODULE 3: UML, ARRAYLIST, STATIC/FINAL, OVERERVING

In de opleiding besteden we veel aandacht aan gestructureerd werken. Daar zijn verschillende methoden voor. Een veelgebruikte ontwerp/modeleringsmethode is UML. Een deel van UML wordt ook in het semester BIS behandeld. In dit semester kijken we naar Class-diagrammen en Sequence-diagrammen.

Daarnaast gaan we ook verder met de technieken achter object georiënteerd programmeren. Hoe verschillend kan het gedrag van objecten zijn en wanneer kunnen we welk concept het beste gebruiken?

## THEORIE

### SCREENCAST ONDERWERP 5: ARRAYLIST

http://www.youtube.com/playlist?list=PLpd9jJvk1PjmJddDbml4Yh_s99TUOJmtx

### READER OVER UML KLASSENDIAGRAMMEN EN SEQUENTIEDIAGRAMMEN

Reader UML class en sequence diagrams (zie bestand op onderwijsonline)

### OPTIONEEL: PLURALSIGHT

UML Class Diagrams:

https://app.pluralsight.com/player?course=uml-introduction&author=mike-erickson&name=uml-introduction-m3-structural-diagrams&clip=1&mode=live

UML Sequence Diagrams:

https://app.pluralsight.com/player?course=uml-introduction&author=mike-erickson&name=uml-introduction-m4-behavioral-diagrams&clip=2&mode=live

Conditional Logic, Looping and Arrays:

https://app.pluralsight.com/player?course=java-fundamentals-language&author=jim-wilson&name=java-fundamentals-language-m4&clip=0&mode=live

### OPTIONEEL: BOEK

#### HOOFDSTUK 4

4.1 tot en met 4.9 pagina 103 tot en met 124 (bovenaan)

4.11 pagina 131 tot en met 134

#### HOOFDSTUK 5

5.6.1 en 5.6.2 pagina 189 tot en met 191

5.7 pagina 193 tot en met 195

## OPGAVE ISBENEDENSCHERM

In de screencast 5.1 over arraylist op tijdstip 8:25 – 8:40 wordt de methode `isBenedenScherm` aan het hoofdprogramma toegevoegd.

Kan deze methode niet beter in de klasse `Deeltje` staan? Leg uit waarom wel of niet. Moet je de methode dan nog aanpassen?

Gebruik in de uitleg het begrip: verbergen van informatie (Boek 5.11 pagina 200 tot en met 203).

## OPGAVE FOR-LUS OM ELEMENTEN TE VERWIJDEREN

### OPGAVE A

Laat zien dat een for-lus die van 0 tot de grootte van de ArrayList loopt niet gebruikt kan worden om elementen te verwijderen.

Gebruik daarvoor het onderstaande programma en teken elke keer dat de tweede for lus doorlopen wordt (regel 8 tot en met 11) het geheugenmodel van de lijst. Zie de reader op #OnderwijsOnline (week 8) om te zien hoe je een ArrayList tekent in het geheugenmodel.

```
public static void main(String[] args) {
     ArrayList<String> lijst = new ArrayList<String>();

     for (int i = 0; i < 4; i++) {
          lijst.add("element: " + i);
     }

     for (int i = 0; i < lijst.size(); i++) {
          String s = lijst.get(i);
          lijst.remove(s);
     }
}
```

### OPGAVE B

Hoe komt het dat in screencast 5.1 op tijdstip 9:00 toch alle Deeltjes uit de ArrayList worden verwijderd?

### OPGAVE C

Maak een programma met for-lus die terugtelt van de grootte van de ArrayList naar 0 en laat zien dat deze lus wel alle elementen uit de ArrayList verwijdert.

## OPGAVE FOREACH VOOR VERWIJDEREN

Maak een kort programmatje waarmee je onderzoekt of je met een foreach lus alle elementen uit een ArrayList kunt verwijderen.

# OEFENINGEN

## OPGAVE LOTTOMACHINE

### FUNCTIONELE SPECIFICATIE

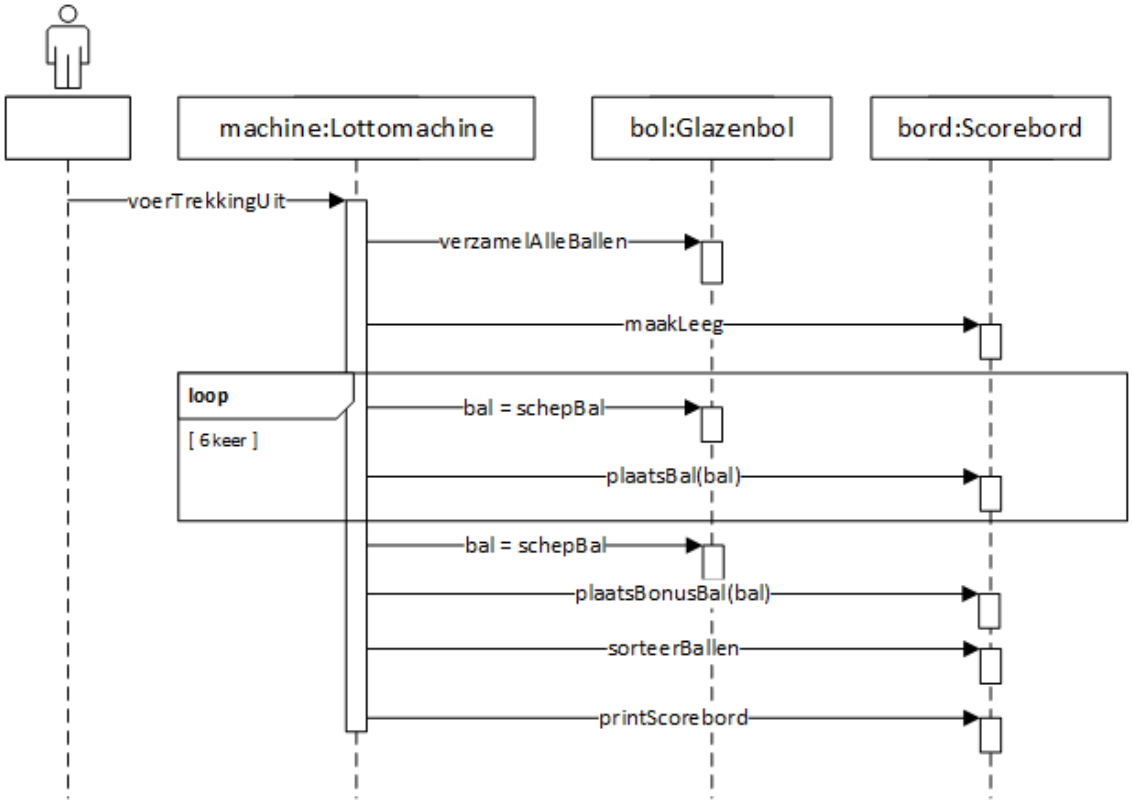
Een lottomachine bestaat uit een glazen bol die in de beginsituatie 45 balletjes, genummerd van 1 tot en met 45, bevat. Verder heeft de lottomachine een 'scorebord' dat uit zeven glazen bestaat. Het besturingsgedeelte van de lottomachine brengt de bol in beweging en zorgt er voor dat er een balletje uit de glazen bol wordt geschept dat in het eerste glas valt. Vervolgens wordt er een tweede balletje uit de bol geschept dat in het tweede glas verdwijnt. Zo worden nog vier balletjes gekozen die in het derde tot en met zesde glas vallen. Het zevende en laatste balletje dat wordt getrokken komt in het zevende glas terecht; het nummer op dit balletje wordt het bonusgetal genoemd. Bij het presenteren van de uitslag worden de eerste zes balletjes die getrokken zijn van klein naar groot getoond.

De uitslag van een lottotrekking wordt bijvoorbeeld als volgt gepresenteerd: 4 11 15 27 31 40
bonusgetal: 18

### TECHNISCHE SPECIFICATIE

De lottoapplicatie gaat uit vier klassen bestaan: Lottomachine, Glazenbol, Scorebord en Lottobal. Een vijfde klasse TestLottoApp bevat de main methode en wordt gebruikt om de machine te testen.

Hieronder is een sequentiediagram te zien waarin de communicatie te zien is tussen de klassen gedurende een trekking. Let op: de methode verzamelAlleBallen is o.a. verantwoordelijk voor het aanmaken van de ballen .




Het poppetje links in het diagram representeert de klasse TestLottoApp.

De implementatie van Lottobal staat hieronder weergegeven:

```
public class Lottobal {

     private int balNummer;

     public Lottobal(int nummer) {
          balNummer = nummer;
     }

     public boolean isNummerGroterDan(Lottobal andereBal) {
          return balNummer > andereBal.balNummer;
     }

     public String toString() {
          return "" + balNummer;
     }
}
```

## OPGAVE A

Maak een klassendiagram met daarin vier klassen (Lottomachine, Glazenbol, Scorebord en Lottobal) waaruit de lottoapplicatie bestaat. Gebruik het sequentiediagram en definitie van Lottobal om te bepalen welke methoden en attributen de klassen moeten hebben.

## OPGAVE B

Maak op basis van het klassendiagram uit opgave A de klassendefinities en implementeer alvast de constructors in elke klasse.

## OPGAVE C

Implementeer de overige methoden in elke klasse door het sequentiediagram van boven naar beneden af te lopen en elke methode te implementeren die je tegen komt.

Ga pas verder met een volgende methode als je de net geïmplementeerde methode getest hebt.

### HINT 1

In de methode schepbal moet je een willekeurige bal selecteren uit een verzameling ballen. Hiervoor heb je de klasse Random nodig. Zie boek 5.4.1 en 5.4.2 (op pagina 182 tot en met pagina 184) voor een beschrijving van deze klasse, of bekijk de Java-API-documentatie.

### HINT 2

Om de ballen op het scorebord te sorteren kun je de onderstaande code gebruiken:

```
     public void sorteerBallen() {
          for (int i = ballen.size(); i > 0; i--) {
               for (int j = 0; j < i-1; j++) {
```

```
                if (ballen.get(j).getNummer() >
ballen.get(j+1).getNummer()) {
                   Lottobal bal = ballen.get(j);
                     ballen.set(j, ballen.get(j + 1));
                   ballen.set(j + 1, bal);
                }
             }
          }
     }
```

## OPGAVE D

In het sorteeralgoritme uit opgave D staat dit statement.

```
if (ballen.get(j).getNummer() > ballen.get(j+1).getNummer())
```

Deze regel zondigt tegen de (zeer zuivere) regel dat een klasse niets mag weten van de interne werking van een andere klasse. In dit geval weet Scorebord dat de klasse Bal ints gebruikt om het nummer bij te houden.

Pas de code in dit if-statement zo aan, dat er weer wordt voldaan aan de bovenstaande regel.

# THEORIE: STATIC & FINAL

## SCREENCAST: STATIC EN FINAL

https://www.youtube.com/playlist?list=PLpd9jJvk1PjkmYIZ40sLe9deC81ILPIAy

## OPTIONEEL: PLURALSIGHT

Static Members, Nested Types and Anonymous Classes:

https://app.pluralsight.com/player?course=java-fundamentals-language&author=jim-wilson&name=java-fundamentals-language-m13&clip=0&mode=live

## OPGAVE SOORTEN VARIABELEN

Gegeven onderstaande 4 begrippen. Zet de begrippen die hetzelfde betekenen bij elkaar.

Klasse variabele, instantievariabele, objectvariabele, statische variabele

## OPGAVE PRINT IN MAIN?

Om niet de hele tijd System.out.println te hoeven typen, wordt de onderstaande code bedacht.

```
public class PrintInMain {

     public static void main(String[] args) {
          print("hallo wereld");
     }

     public void print(String tekst) {
          System.out.println(tekst);
     }
}
```

Op regel 4 treedt dan de volgende foutmelding op.

*Cannot make a static reference to the non-static method print(String) from the type PrintInMain*

## OPGAVE A

Leg uit wat er met deze foutmelding bedoeld wordt.

## OPGAVE B

Implementeer twee verschillende oplossingen voor deze foutmelding.

## OPGAVE C

Welke oplossing verdient volgens jou de voorkeur.

## OPGAVE STUDENT UITBREIDEN

Gegeven de code voor een student uit de screencast

```
public class Student {
	private String naam;
	private String geslacht;

	public static final String MAN = "man";
	public static final String VROUW = "vrouw";

	private static int nStudenten = 0;

	public Student(String naam) {
		this.naam = naam;
		nStudenten++;
	}

	public String getGeslacht() {
		return geslacht;
	}

	public void setGeslacht(String geslacht) {
		this.geslacht = geslacht;
	}

	public static int getNStudenten() {
		return nStudenten;
	}

	public String toString() {
		return getNaam();
	}

	public String getNaam() {
		return naam;
	}

	public void setNaam(String naam) {
		this.naam = naam;
	}
}
```

### OPGAVE A

Teken het geheugenmodel van onderstaande code op het moment dat het programma net voorbij regel 3 is. Bedenk met name waar de variabele nStudenten te vinden is.

```
public class DemoApp {
	public static void main(String[] args) {
		Student s = new Student("han");
		System.out.println(s.getNaam());
	}
}
```

### OPGAVE B

Maak een nieuwe instantievariabele nummer die het studentennummer van een student kan bijhouden.

### OPGAVE C

Hoewel er constanten zijn gemaakt, kan een gebruiker van de klasse Student nog steeds een geslacht meegeven met een andere waarde als “man”, of “vrouw”. Los dit probleem op.

## NIET VERPLICHTE UITDAGING: ENUMS

Het gebruik van constanten zoals dat in de klasse Student te zien is, kom je vaker tegen. Toch is er een betere manier om ervoor te zorgen dat het geslacht van een student slechts twee voor gedefinieerde waarden kan hebben. Daarvoor heb je een zogenaamde Enum nodig.

### OPGAVE A

Zoek uit hoe je een Enum kunt gebruiken in Java en pas de klasse Student zo aan, dat er een Enum gebruikt wordt voor het geslacht.

### OPGAVE B

Noem een voordeel van het gebruik van een Enum boven het gebruik van constanten.

# OEFENINGEN

## OPGAVE STATIC ABC

Gegeven onderstaande 3 klassen (A, B en C):

```
public class A {

	private static String a = "a";

	public static String getA() {
		return a;
	}
}

public class B {

	private static String b = "b";

	public String getB() {
		return b;
	}
}

public class C {

	private String c = "c";

	public static String getC() {
		return c;
	}
}
```

### OPGAVE A

Welke klasse (A, B, of C) compileert niet?

### OPGAVE B

Leg uit wat de reden voor deze foutmelding is?

## OPGAVE STUDIEADVIES I

Gegeven de klasse Student. Een student heeft een naam en een cijferlijst met acht cijfers. Voor het gemak zijn de cijfers integers.

```
public class Student {
	private String naam;
	private int[] cijfers;

	public Student(String naam) {
		this.naam = naam;
		cijfers = new int[8];
	}

	public void setCijfer(int vaknummer, int cijfer) {
		cijfers[vaknummer] = cijfer;
	}

	public int[] getCijfers() {
		return cijfers;
	}

	public String toString() {
		String representatie = "naam: " + naam + "\ncijfers: ";
		for (int cijfer : cijfers) {
			representatie += " " + cijfer;
		}
		return representatie;
	}
}
```

De klasse Studieadviseur krijgt de verantwoordelijkheid te bepalen of een student een positief studieadvies krijgt.

Een student krijgt een positief studieadvies als er ten minste vier cijfers zijn die groter of gelijk zijn aan een 6.

Hieronder is de definitie van de klasse te vinden:

```
public class Studieadviseur {

	public static boolean krijgtPositiefStudieadvies(Student s) {
		//Implementatie moet je zelf maken, zie opgave
	}
```

```
}
```

Deze klasse kan met onderstaande code getest worden:

```
public class TestStudieadviesApp {

	public static void main(String[] args) {
		Random r = new Random();

		Student s1 = new Student("persoon 1");
		for (int i = 0; i < 8; i++) {
			s1.setCijfer(i, r.nextInt(10) + 1);
		}

		System.out.println(s1);

	System.out.println(Studieadviseur.krijgtPositiefStudieadvies(s1));
		System.out.println("------------------");
	}
}
```

### OPGAVE A

Implementeer de methode krijgtPositiefStudieAdvies in de klasse Studieadviseur.

### OPGAVE B

Geef een reden waarom het een goede keuze is om de methode krijgtPositiefStudieadvies static te maken.

### OPGAVE C

Het zou kunnen zijn dat de keuze om bovenstaande methode static te maken, niet in elke situatie een bruikbare keuze is. Verzin een situatie waarin de methode krijgtPositiefStudieadvies beter niet static kan zijn.

## OPGAVE STUDIEADVIES II

Er wordt een klasse gemaakt waarmee studenten gegenereerd kunnen worden die een willekeurige cijferlijst hebben:

```
public class RandomStudentenGenerator {

	private Student[] studentenLijst;

	public static void genereerStudenten(int aantal) {
		Random generator = new Random();
		studentenLijst = new Student[aantal];
		for (int i = 0; i < aantal; i++) {
```

```
                studentenLijst[i] = new Student("persoon_" + i);

                for (int j = 0; j < 8; j++) {
                     studentenLijst[i].setCijfer(j, generator.nextInt(10) +
1);
                }
           }
     }

     public Student[] getStudentenLijst() {
          return studentenLijst;
     }
}
```

### OPGAVE A

Plaats de code in Eclipse en je krijgt (de bekende) foutmelding: *Cannot make a static reference to the non-static field studentenLijst*.

Leg uit wat er mis is.

### OPGAVE B

Er zijn drie mogelijke oplossingen voor bovenstaande foutmelding:

#### OPLOSSING I: STATIC TOEVOEGEN OP TWEE PLEKKEN

```
public class RandomStudentenGenerator {

     private static Student[] studentenLijst; // hier static toevoegen

     public static void genereerStudenten(int aantal) {
          Random generator = new Random();
          studentenLijst = new Student[aantal];
          for (int i = 0; i < aantal; i++) {
               studentenLijst[i] = new Student("persoon_" + i);

               for (int j = 0; j < 8; j++) {
                    studentenLijst[i].setCijfer(j, generator.nextInt(10) +
1);
               }
          }
     }

     public static Student[] getStudentenLijst() { // hier static toevoegen
```

```
          return studentenLijst;
     }
}
```

### OPLOSSING II: STATIC VERWIJDEREN BIJ GENEREERSTUDENTEN

```
public class RandomStudentenGenerator {

     private Student[] studentenLijst;

     public static void genereerStudenten(int aantal) { // hier static
verwijderen
          Random generator = new Random();
          studentenLijst = new Student[aantal];
          for (int i = 0; i < aantal; i++) {
               studentenLijst[i] = new Student("persoon_" + i);

               for (int j = 0; j < 8; j++) {
                    studentenLijst[i].setCijfer(j, generator.nextInt(10) +
1);
               }
          }
     }

     public Student[] getStudentenLijst() {
          return studentenLijst;
     }
}
```

### OPLOSSING III: INSTANTIEVARIABELE VERWIJDEREN EN RETURN STATEMENT TOEVOEGEN

```
public class RandomStudentenGenerator {

    private Student[] studentenLijst;

    public static Student[] genereerStudenten(int aantal) {
        Random generator = new Random();
        Student[] studentenLijst = new Student[aantal];
        for (int i = 0; i < aantal; i++) {
            studentenLijst[i] = new Student("persoon_" + i);

            for (int j = 0; j < 8; j++) {
                studentenLijst[i].setCijfer(j, generator.nextInt(10) +
1);
            }
        }
        return studentenLijst;
    }

    public Student[] getStudentenLijst() {
        return studentenLijst;
    }

}
```

Leg uit welke oplossing je het meest aantrekkelijk vindt.

## UITDAGING VERKIEZINGSUITSLAG

Bij verkiezingen voor de Tweede Kamer wordt de zetelverdeling berekend met behulp van een bijzondere procedure, waarbij het vooral gaat om de verdeling van de zogenaamde restzetels. De procedure gaat zo, met voorbeelden uit de uitslag van de verkiezingen van 2010 (de werkelijkheid is natuurlijk inmiddels anders):

- Men berekent eerst het totale aantal stemmen (2010: 9.416.001) en deelt dit door 150, dit is de kiesdeler (2010: 62.773).

- Elke partij krijgt een aantal zetels door het stemmenaantal van deze partij te delen door de kiesdeler en af te ronden naar beneden. De VVD kreeg in 2010 zo 30 volle zetels

- Sommige partijen hebben minder stemmen dan de kiesdeler, ze 'halen de kiesdrempel niet'. Zij krijgen geen zetels en doen verder ook niet mee in de verdeling van de restzetels.

- Omdat er naar beneden afgerond wordt en partijen de kiesdrempel niet halen, worden zo minder dan 150 zetels vergeven. De andere zetels - de restzetels - worden als volgt verdeeld:

o Voor elke partij deelt men het aantal stemmen door het aantal zetels plus 1 (voor de VVD bereken je dus 1929575 / 31).

o De partij met het grootste aantal stemmen per zetel krijgt een restzetel. (In 2010 kreeg de VVD de eerste restzetel) Voor die partij bereken je het aantal stemmen per zetel opnieuw (VVD: nu dus 1929575 / 32)

o Je herhaalt de vorige stap totdat alle restzetels vergeven zijn.

Let op! Het is niet goed genoeg om in één keer de volgorde van de aantallen stemmen per zetel af te gaan. Het kan namelijk gebeuren dat een grote partij eerder een tweede restzetel krijgt dan een andere, kleine partij. (Na het toekennen van de eerste restzetel daalt het aantal stemmen per zetel van de VVD tot ongeveer 60300, daardoor zou de VVD een tweede restzetel krijgen voor D66 de eerste krijgt.)

NB: De Nederlandse kieswet staat lijstverbindingen toe, waarbij twee of meer partijen bij elkaar genomen worden bij het verdelen van de restzetels. We laten deze buiten beschouwing.

Schrijf een programma dat de zetelverdeling berekent uitgaande van de aantallen stemmen per partij. Maak daarbij een ArrayList van 'PartijUitslagen', waarop je de procedure voor de zetelverdeling bij schrijft. Je hierbij gebruik maken van de startcode die op OnderwijsOnline staat: "Verkiezingsuitslagen.zip". De startcode bevat de uitslagen van vijf verkiezingen (periode 1998-2010).

| Partij | Stemmen | Percentage | Zetels basic | Restzetels | Stemmen per (ztl+1) |
|---|---|---|---|---|---|
| VVD | 1.929.575 | 20,49 | 30 | 0 | 62244,35 |
| PvdA | 1.848.805 | 19,63 | 29 | 0 | 61626,83 |
| PVV | 1.454.493 | 15,45 | 23 | 0 | 60603,88 |
| CDA | 1.281.886 | 13,61 | 20 | 0 | 61042,19 |
| SP | 924.696 | 9,82 | 14 | 0 | 61646,40 |
| D66 | 654.167 | 6,95 | 10 | 0 | 59469,73 |
| GroenLinks | 628.096 | 6,67 | 10 | 0 | 57099,64 |
| ChristenUnie | 305.094 | 3,24 | 4 | 0 | 61018,80 |
| SGP | 163.581 | 1,74 | 2 | 0 | 54527,00 |
| PvdD | 122.317 | 1,30 | 1 | 0 | 61158,50 |
| Trots op NL | 52.937 | 0,56 | 0 | 0 | NVT |
| Mens en Spirit | 26.196 | 0,28 | 0 | 0 | NVT |
| Piratenpartij | 10.471 | 0,11 | 0 | 0 | NVT |
| Lijst 17 | 7.456 | 0,08 | 0 | 0 | NVT |
| Partij één | 2.042 | 0,02 | 0 | 0 | NVT |
| Nieuw NL | 2.010 | 0,02 | 0 | 0 | NVT |
| Heel NL | 1.255 | 0,01 | 0 | 0 | NVT |
| EPN | 924 | 0,01 | 0 | 0 | NVT |
| | | | | | |
| Totaal | 9.416.001 | Totaal base zetels | 143 | | |
| Kiesdeler | 62.773 | Restzetels | 7 | | |

# OVERERVING

## THEORIE

### SCREENCAST: OVERERVING

http://www.youtube.com/playlist?list=PLpd9jJvk1PjmlCr4hMlUwdLF9nJNtYVwv

### OPTIONEEL: PLURALSIGHT

Class Inheritance

https://app.pluralsight.com/player?course=java-fundamentals-language&author=jim-wilson&name=java-fundamentals-language-m8&clip=0&mode=live

### OPTIONEEL: BOEK

Hoofdstuk 8

8.1 t/m 8.6 pagina 296 t/m 318

Hoofdstuk 9

9.5 pagina 342

9.9 pagina 349 en 350

## OPGAVE OVERERVINGSHIËRARCHIE

### OPGAVE A

Teken een overervingshiërarchie waarin de onderstaande klassen elk één keer voorkomen (Kiwi staat er dus drie keer in).

Dier, Vogel, Levensvorm, Vrucht, Mens, Kiwi, Kiwi, Kiwi.

Hint:

**Afbeelding 2 Kiwi (Wikimedia commons, 2015a)**

**Afbeelding 1 Kiwi (Wikimedia commons, 2015b)**

**Afbeelding 3 Kiwi (John Key, premier Nieuw-Zeeland) (Wikimedia commons, 2015c)**

### OPGAVE B

Implementeer de klassen Vogel, Vrucht, Kiwi en Kiwi en los het naamconflict dat je krijgt op zonder de klassennamen van beid Kiwi's aan te passen (zie eventueel boek pagina 188).

## OPGAVE SUPERCONSTRUCTOR

In deze opgave maken we alleen gebruik van de klasse Dier en de klasse Kiwi.

### OPGAVE A

Bekijk de constructor van de klasse Kiwi. Wanneer deze constructor wordt uitgevoerd, wordt eerst de constructor van dier uitgevoerd. Geef de header (boek pagina 36) van deze constructor.

```
public class Dier {
     protected String naam;
}


public class Kiwi extends Dier {
     private int loopSnelheid;

     public Kiwi(String naam, int loopSnelheid) {
          this.loopSnelheid = loopSnelheid;
     }
}
```

### OPGAVE B

De constructor die je bij opgave A hebt genoemd is niet expliciet gedefinieerd in de klasse Dier. Beschrijf zo exact mogelijk de spelregel die maakt dat deze constructor toch wordt uitgevoerd.

### OPGAVE C

Er wordt een constructor toegevoegd aan de klasse Dier:

```
public class Dier {
     protected String naam;

     public Dier(String naam) {
          this.naam = naam;
     }
}
```

Nu geeft de constructor van Kiwi (regel 4) de foutmelding

*Implicit super constructor Dier() is undefined. Must explicitly invoke another constructor*

```
public class Kiwi extends Dier {
     private int loopSnelheid;

     public Kiwi(String naam, int loopSnelheid) {
```

```
            this.loopSnelheid = loopSnelheid;
      }
}
```

Leg uit wat deze foutmelding betekent.

### OPGAVE D

De foutmelding uit opgave C kun je oplossen door onderstaande code:

```
public class Kiwi extends Dier {
      private int loopSnelheid;

      public Kiwi(String naam, int loopSnelheid) {
            super(naam);
            this.loopSnelheid = loopSnelheid;
      }
}
```

Onderzoek of je regels 5 en 6 ook mag omdraaien, zonder een foutmelding te krijgen.

## BRONNEN

Wikimedia commons, 2015a. "Apteryx owenii 1" by G.D. Rowley - Rowley, G.D., Ornithological Miscellany, 1875-78 - http://www.nzbirds.com/birds/kiwils.html. Licensed under Publiek domein via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Apteryx_owenii_1.jpg#/media/File:Apteryx_owenii_1.jpg

Wikimedia commons, 2015b. "Kiwi (Actinidia chinensis) 1 Luc Viatour" by Luc Viatour - own work www.lucnix.beNikon case D300 optical Sigma 150mm F2,8 macro. Licensed under GFDL via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Kiwi_(Actinidia_chinensis)_1_Luc_Viatour.jpg#/media/File:Kiwi_(Actinidia_chinensis)_1_Luc_Viatour.jpg

Wikimedia commons, 2015c, "John Key National Party2" by Guo's - cropped from 2008-3-19 Mr John Key and Me.. Licensed under CC BY-SA 2.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:John_Key_National_Party2.jpg#/media/File:John_Key_National_Party2.jpg

# OEFENINGEN

## OPGAVE VIERKANT EN RECHTHOEK

Wat is de overervingsrelatie tussen een vierkant en een rechthoek. Erft een rechthoek van een vierkant of erft een vierkant van een rechthoek? Beargumenteer je keuze.

## OPGAVE ABC

Gegeven onderstaande klassen:

```
public class A {
	protected String sa;

	public A(String s1) {
		this.sa = s1;
	}
}

public class B extends A {
	protected String sb;

	public B(String s1, String s2) {
		super(s1);
		sb = s2;
	}
}

public class C extends B {
	protected String sc;

	public C(String s1, String s2, String s3) {
		super(s1, s2);
		sc = s3;
	}
}
```

En de klasse met het hoofdprogramma:

```
public class ABC {
	public static void main(String[] args) {
		C c = new C("a", "b", "c");
	}
}
```

### OPGAVE A

Teken het geheugenmodel van dit programma op het moment dat de in het hoofdprogramma de constructor van c is aangeroepen, in klasse C de aanroep van super wordt gedaan, in klasse B de aanroep van super wordt gedaan en in klasse A de eerste regel van de constructor net is uitgevoerd. Zie ook de highlights in de code.

Teken in het geheugenmodel alle stack-frames vanaf de aanroep new C(); en laat alleen objecten van klasse A, B en C zien.

## OPGAVE B

Vervang in de klassen A, B en C de naam van de instantievariabelen sa, sb en sc in s. Pas ook de drie constructors aan.

Voeg drie println statements toe aan de constructor van de klasse C waarmee je de waarden van alle drie de variabelen s in de console laat zien. Lukt dit?

## OPGAVE FIGUREN 1

Gegeven de klassen Cirkel en Rechthoek:

```
import processing.core.PApplet;

public class Cirkel {

     private float x, y, vx, vy, ax, ay;
     private float diameter;
     private int kleur;

     public Cirkel(float x, float y, float diameter) {
          this.x = x;
          this.y = y;
          this.diameter = diameter;
          zetStil();
          kleur = 0xFFFFFFFF;
     }

     public void doeStap() {
          if (!staatStil()) {
               pasVersnellingToe();
               pasSnelheidToe();
          }
     }

     public void setSnelheid(float vx, float vy) {
          this.vx = vx;
          this.vy = vy;
     }

     public void setVersnelling(float ax, float ay) {
          this.ax = ax;
          this.ay = ay;
     }

     public void zetStil() {
          vx = vy = ax = ay = 0;
     }

     public boolean staatStil() {
          return (vx == 0 && vy == 0 && ax == 0 && ay == 0);
     }

     public void tekenCirkel(PApplet p) {
```

```
          p.noStroke();
          p.fill(kleur);
          p.ellipse(x, y, diameter, diameter);
     }

     public void setKleur(int kleur) {
          this.kleur = kleur;
     }

     private void pasVersnellingToe() {
          vx += ax;
          vy += ay;
     }

     private void pasSnelheidToe() {
          x += vx;
          y += vy;
     }
}

import processing.core.PApplet;

public class Rechthoek {

     private float x, y, vx, vy, ax, ay;
     private float breedte, hoogte;
     private int kleur;

     public Rechthoek(float x, float y, float breedte, float hoogte) {
          this.x = x;
          this.y = y;
          this.breedte = breedte;
          this.hoogte = hoogte;
          zetStil();
          kleur = 0xFFFFFFFF;
     }

     public void doeStap() {
          if (!staatStil()) {
               pasVersnellingToe();
               pasSnelheidToe();
          }
     }

     public void setSnelheid(float vx, float vy) {
          this.vx = vx;
          this.vy = vy;
     }

     public void setVersnelling(float ax, float ay) {
          this.ax = ax;
          this.ay = ay;
     }
```

```
    public void zetStil() {
        vx = vy = ax = ay = 0;
    }

    public boolean staatStil() {
        return (vx == 0 && vy == 0 && ax == 0 && ay == 0);
    }

    public void tekenRechthoek(PApplet p) {
        p.noStroke();
        p.fill(kleur);
        p.rect(x, y, breedte, hoogte);
    }

    public void setKleur(int kleur) {
        this.kleur = kleur;
    }

    private void pasVersnellingToe() {
        vx += ax;
        vy += ay;
    }

    private void pasSnelheidToe() {
        x += vx;
        y += vy;
    }
}
```

## OPGAVE A

Maak een hoofdprogramma waarin je alle publieke methoden van Cirkel en Rechthoek test.

## OPGAVE B

Verwijder alle gedupliceerde code door gebruik te maken van overerving. Pas de code in het hoofdprogramma niet aan en test of alles nog steeds werkt.

## OPGAVE C

Voeg aan de superklasse die je in opgave B hebt gemaakt een variabele toe waarmee je kunt bijhouden of de figuur zichtbaar is of onzichtbaar. Maak een getter en setter voor deze variabele en gebruik deze variabele om de instanties van Rechthoek en Cirkel zichtbaar en onzichtbaar te maken. Test deze nieuwe mogelijkheid in het hoofdprogramma.

# STATISCH & DYNAMISCH

## THEORIE

### Screencast: statische en dynamische types en abstract

http://www.youtube.com/playlist?list=PLpd9jJvk1PjmbJnRN4kOfpYgKvJ2PCDsP

#### OPTIONEEL: PLURALSIGHT

Static Members, Nested Types and Anonymous Classes:

https://app.pluralsight.com/player?course=java-fundamentals-language&author=jim-wilson&name=java-fundamentals-language-m13&clip=0&mode=live

#### BOEK

##### HOOFDSTUK 8

8.7 t/m 8.11 pagina 319 t/m 328

##### HOOFDSTUK 9

9.1 t/m 9.4, pagina 332 t/m 341

9.6 t/m 9.12 pagina 343 t/m 356

##### HOOFDSTUK 10

10.3 en 10.4 pagina 376 t/m 385 details van de code zijn niet belangrijk, zorg dat je de definities snapt

## OPGAVE COMPILE-TIME VS RUNTIME

Hieronder is een lijst van begrippen te vinden. Geef per begrip aan of ze gedurende compile-time of gedurende runtime een rol spelen.

- Abstract
- Klasse
- Instantie
- Dynamische type
- Statische type
- Geheugenmodel
- Methode look-up
- Uitvoeren van een programma
- Controleren van een programma

## OPGAVE GEHEUGENMODEL KNOP

Hieronder is een gedeelte van een hoofdprogramma gegeven.

```
1   @SuppressWarnings("serial")
2   public class KnopApp extends PApplet {
        ...
3       private Licht l;
4       private Knop k;

5       public void setup() {
6           l = new Licht(this);
7           k = new LichtKnop(this, l, 20, 20, 50, 50);
        ...
8       }
        ...
9   }
10
11
12
13
```

Maak een geheugenmodel van het programma op het moment dat het statement op regel 9 (vetgedrukt) net is uitgevoerd. Je hoeft de velden uit LichtKnop niet weer te geven en voor de klassen Knop, LichtKnop en Licht wordt de code uit de screencasts gebruikt.

## OPGAVE ABC 2

Gegeven onderstaande klassen:

```
public class A {

    @Override
    public String toString() {
        return "A";
    }
}


public class B extends A {

    @Override
    public String toString() {
        return "B";
    }
}


public class C extends B {

    @Override
    public String toString() {
```

```
        return "C";
    }
}
```

## OPGAVE A

In de klasse A staat de annotatie @Override bij de methode toString waarmee gesuggereerd wordt dat deze methode in een superklasse van A bestaat en overschreven wordt. Erft de klasse A inderdaad van een andere klasse die de methode toString definieert, of kan de annotatie beter weggelaten worden?

## OPGAVE B

Gegeven onderstaande hoofdprogramma:

```
1  public class ABC2 {
2      public static void main(String[] args) {
3          A a1 = new A();
4          A a2 = new B();
5          A a3 = new C();
6          B b1 = new A();
7          B b2 = new B();
8          B b3 = new C();
9          C c1 = new A();
10         C c2 = new B();
11         C c3 = new C();
12
13         System.out.println(a1.toString());
14         System.out.println(a2.toString());
15         System.out.println(a3.toString());
16         System.out.println(b1.toString());
17         System.out.println(b2.toString());
18         System.out.println(b3.toString());
19         System.out.println(c1.toString());
20         System.out.println(c2.toString());
21         System.out.println(c3.toString());
22     }
23 }
```

Geef voor elk statement op regel 3 t/m 11 aan of ze door de compiler heenkomen, of dat er een foutmelding optreedt. Welk patroon zie je?

## OPGAVE C

Geef voor elk statement op regel 13 t/m 21 aan uit welke klasse de methode toString komt die tijdens het uitvoeren van het programma gebruikt wordt. Sla uiteraard de regels over die vanwege eerdere compileerfouten niet kunnen worden uitgevoerd.

# OEFENINGEN

## OPGAVE CASTEN MET ABC 2

De klassen A, B en C zijn uitgebreid zoals hieronder is weergegeven.

```
public class A {

	@Override
	public String toString() {
		return "A";
	}

	public void doA() {
		System.out.println("AA");
	}
}

public class B extends A {

	@Override
	public String toString() {
		return "B";
	}

	public void doB() {
		System.out.println("BB");
	}
}


public class C extends B {

	@Override
	public String toString() {
		return "C";
	}

	public void doC() {
		System.out.println("CC");
	}
}
```

## OPGAVE A

Onderzoek in het programma ABC2 de of de onderstaande casts mogelijk zijn zonder een error op te leveren.

- a3 casten naar C
- a2 casten naar C
- a1 casten naar C

- b2 casten naar A
- c3 casten naar B
- c3 casten naar A

Ga als volgt te werk:

Implementeer de cast in het hoofdprogramma en controleer welke van de drie methoden doA, doB en doC eclipse je nu toestaat aan te roepen.

Run vervolgens het programma en controleer of je nu een runtime error krijgt.

### OPGAVE B

Welke algemene conclusies kun je trekken op basis van dit experiment?

## OPGAVE FIGUREN 2

*Vervolg van Opgave figuren. Nu voegen we een abstracte klasse toe zoals ook in de screencast te zien is.*

### OPGAVE A

In de opgave Figuren 1 uit week 8 heb je de superklasse Figuur gemaakt om alle gedupliceerde code uit rechthoek en cirkel te verhelpen. Voeg aan deze klasse de abstracte methode teken toe en maak de klasse zelf ook abstract.

### OPGAVE B

Vervang in Cirkel en Rechthoek de methoden tekenCirkel en tekenRechthoek door de methode teken en zorg dat dit een Override is van de abstracte methode uit de klasse Figuur.

### OPGAVE C

Niet elk figuur hoeft te kunnen bewegen en daarom zou je kunnen overwegen om alle code die te maken heeft met snelheid en versnelling uit Figuur te halen en in een nieuwe klasse BeweegbaarFiguur te stoppen. Wanneer je dit doet, dan zou je ook een BeweegbareCirkel en een BeweegbareRechthoek moeten maken.

Teken een klassendiagram met Figuur, BeweegbaarFiguur, Cirkel, Rechthoek, BeweegbareCirkel, BeweegbareRechthoek. Van alle velden en methoden hoef je alleen de methoden public void setKleur(int kleur), public void doeStap(), public void setSnelheid() en public void teken(PApplet p) op te nemen. Geef wel duidelijk aan welke methoden abstract zijn en welke niet.

### OPGAVE D

De indeling die je in opgave C hebt getekend zorgt voor gedupliceerde code. Welke code ben je aan het dupliceren?

### OPGAVE E

Zou je op basis van opgave C en D adviseren een onderscheid te maken in Figuur en BeweegbaarFiguur, of zou je alleen de klasse Figuur die ook alle code bevat om figuren te laten bewegen? Beargumenteer het antwoord.