**Algemene informatie Basis Programmeren**

Krijn Hoogendorp

ROCvA 2016-2017

Versie 0.9 (juli 2016)

1. **Licentie**

Deze cursus maakt gebruik van het materiaal “Algorithmic Thinking and Structured Programming (in Greenfoot)”[[1]](#footnote-1) van Renske Smetsers-Weeda & Sjaak Smetsers. Dit materiaal is gelicenseerd als Creative Commons Attribution 4.0[[2]](#footnote-2).

1. **Leerdoelen**

Na deze cursus heeft/kan de leerling/student:

* basiskennis van logica;
* logica toepassen op problemen;
* enkele klassieke algoritme problemen herkennen;
* algoritmes ontwikkelen ;
* onderscheiden van functioneel, imperatief en object-georiënteerd programmeren
* kunnen toepassen van de volgende concepten in de programmeertaal Java:

1. datatypes: integer, float, boolean, character, String.
2. Gebruik van variabelen in programmeren
3. Basic operators: +, -, /, \*, modulus, ++, --
4. Relational operators: ==, !=, > , <, <=, >=
5. Logical operators: and, or, not, xor
6. iteratie
7. If-then statement
8. Loops: while, for
9. Arrays/Lists
10. **Inhoud**

De cursus bestaat uit 7 opgaven. Deze hebben de volgende onderwerpen:

1. Kennismaken met Greenfoot: methods, objects, classes
2. Syntax
3. Optimaliseren van oplossingen: submethods, conditions. nesting
4. Generieke oplossingen (logial operators, return statements, condities)
5. Variabelen, initialiseren, operators, while-loop
6. Constructors, if-then statement,
7. Objecten, for-each loop, Java library
8. **Inzet**

Je kan deze cursus niet goed volbrengen door alleen in de klas aan de opdrachten te werken. Er is per week 1,5 uur les. Daarnaast wordt verwacht dat er minstens 1,5 uur p/w thuis aan wordt gewerkt.

**Opdracht 1: Maak kennis met Dodo**

1. **Inleiding**

In deze opgavenreeks Objectgeorienteerd programmeren in Greenfoot kun je op een leuke manier de basisvaardigheden leren voor Objectgeorienteerd programmeren. Na afloop kun je dan, met behulp van de Greenfoot bibliotheken, programmeren in Java.

Met deze basisvaardigheden ben je ook in staat om makkelijk andere Objectgeorienteerde talen leren. Je leert gebruik maken van voorgegeven stukken programmacode en deze uit te breiden waar nodig. Hergebruik van bestaande

programmacode is een van de krachten van Java.

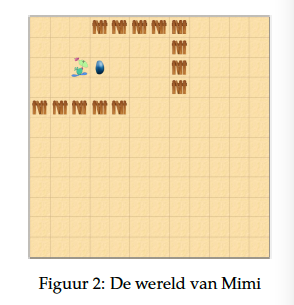
Je programmeert in de taal Java. In de opdrachten werken we met de Greenfoot programmeeromgeving. Dit is een omgeving die je helpt Java programma’s te schrijven zonder eerst uitgebreid de theorie van programmeren te leren. Zo kun je gaandeweg Java leren terwijl je er mee bezig bent.

1. **Uitleg: Wegwijs in Greenfoot**

Dit is onze Dodo. Ze heet Mimi en behoort tot de Dodo familie.



Mimi leeft in een wereld die bestaat uit 12 bij 12 vakjes. Haar levensdoel is het leggen en uitbroeden van eieren. In het plaatje hieronder zie je Mimi in de wereld met een ei voor haar. In enkele vakken staan hekjes.



Je kan Mimi opdrachten laten uitvoeren, zoals:

- move stap een hokje verder

- hatchEgg broed een ei uit

- jump sprin een aantal vakjes vooruit

Het is ook mogelijk om Mimi een vraag te stellen:

- canMove Kun je een stapje opzij zetten?

- getNrOfEggsHatched Hoeveel eieren heb je uitgebroed?

1. **Aan de slag met Greenfoot**

Download Greenfoot op <http://www.greenfoot.org/download> en volg de aangegeven instructies.

Via de mail het je het scenario ‘MadagaskerOpdr1’ gekregen. Download dit op je computer en open het.

Druk rechtsonder op de knop ‘Compile’ . Als het goed is krijg je het volgende te zien:



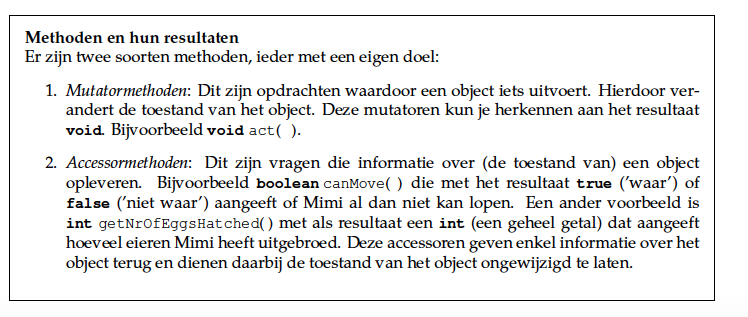
1. **Maak objecten**

De wereld in Greenfoot bestaat uit objecten. In onderstaand figuur zie je één *Dodo* object en drie *Egg* objecten. Het Dodo object maak je als volgt:

1. klik met de rechtermuisknop op MyDodo, rechts in het scherm.
2. Kies *new MyDodo()* om een Dodo object te maken.
3. Sleep het object, de MyDodo, naar een vakje in de wereld
4. Leg op dezelfde manier de drie eieren naar. Tip: houd ‘Shift’ ingedrukt om meerdere eieren neer te leggen.
5. **Methoden**

Beantwoord de volgende vragen:

1. Greenfoot heeft een aantal knoppen onderaan het venster. Wat gebeurt er als je op de ‘Act’ knop klikt? **(de dodo draaid)**
2. Wat gebeurt er als je nog eens op ‘Act’’ klikt? En nog eens? **(de dodo beweegt)**
3. Wat gebeurt er als Mimi op verschillende plekken staat (je kan haar verplaatsen m.b.v. de muis). Wat gebeurt er als Mimi aan de rand van de wereld staat? **(draaid en gaat dan met de hoek mee)**
4. Beschrijf in eigen woorden wat de ‘Act’ knop doet **(het standaard loop programme word geactieveerd)**
5. Druk op de ‘Run’ knop. Wat gebeurt er? **(het loop programma word geactieveerd)**
6. Wat hebben ‘Act’ en ‘Run’ met elkaar te maken? **(Act laat het loop programme 1 keer gaan , Run zorgt er voor dat er eigenlijk de heletijd op act word gedrukt)**
7. Wat de knop ‘Reset’ **(alles word gerest en je krijgt 12 legen vakjes te zien)**



Mutatormethodes

Onze MyDodo is geprogrammeerd om een aantal dingen te doen. Met de mutatormethodes kan Mimi de volgende opdrachten uitvoeren (oefen met deze opdrachten):

1. Klik met de rechtermuisknop op Mimi. Zo zie je wat Mimi allemaal kan (welke methodes je kan aanropen), bijvoorbeeld act(), move() en hatchEgg().
2. Roep act() aan. Wat doet die? **(Mini loopt 2 vakjes)**
3. Kijk naar jouw beschrijving van wat de ‘Act’ knop doet (uit vorige opgave).
4. Zou ere en verschil zijn tussen het klikken op de ‘Act’ knop en het aanroepen van een method? (**Act is specifiek voor lopen)**
5. Leg een ei neer in de wereld. Sleep Mimi op het ei. Zet je muis op Mimi (niet op het ei). Roep met de rechtermuisknop de method void hatchEgg() aan . Wat gebeurt er? **(Mini ‘broed’ het ei uit (ei gaat weg) )**

Accessormethodes

Met accessormethodes kan Mimi vragen beantwoorden waarmee ze informative over haarzelf geeft.

1. Mimi kan lopen (maar niet vliegen). Als Mimi in het midden van de wereld staat en je vraagt haar of ze een stap kan zetten, wat krijg je dan als antwoord? En wat is het antwoord als Mimi in een hoek staat?

**(als Mini in het midden staat een een stap doet loopt ze gewoon de richting naar waarop ze kijkt)**

**(als Mini in een hoek staat en tegen een muur kijkt komt er een “I am stuck” pop-up , als ze niet tegen een muur kijkt loopt ze verder)**

1. Roep de methode Boolean canMove() aan. Welk antwoord krijg je terug?



1. Verplaats Mimi naar de rand (met haar snavel naar de rand gericht). Roep boolean canMove() aan Wat is nu het antwoord **(False)**

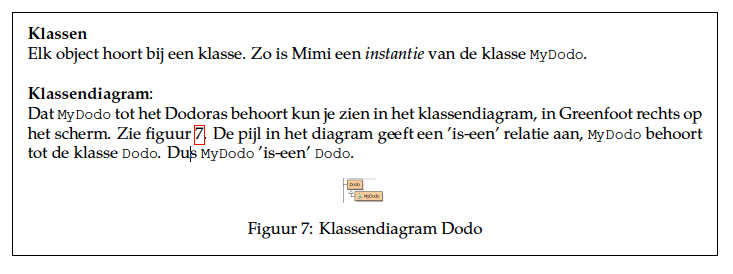


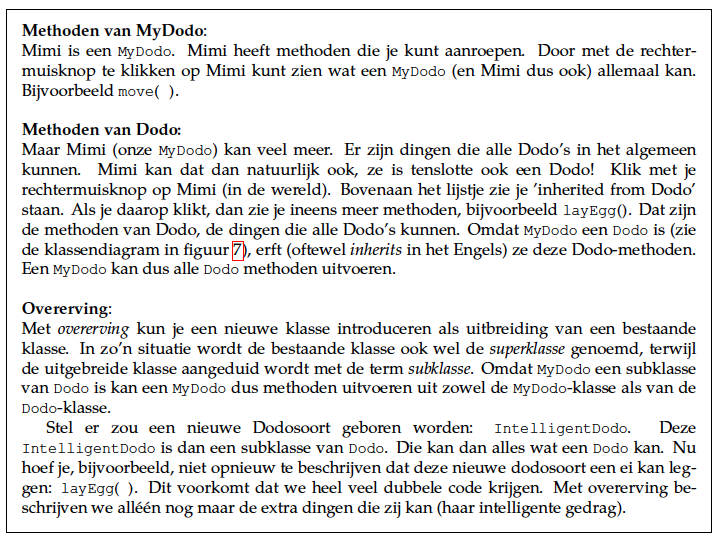
1. Roep nu de method int getNrOfEggsHatched() aan. Wat is het antwoord? **(0)**

1. De int getNrOfEggsHatched() method kun je gebruiken om te zien hoeveel eieren MyDodo heeft uitgebroed. Kun je nog een ei neerzetten en uitbroeden (met hatchEgg()) en daarna kijken met getNrOfEggsHatched() of het gelukt is?

**(Ja het is gelukt)**

**6. Overerving**

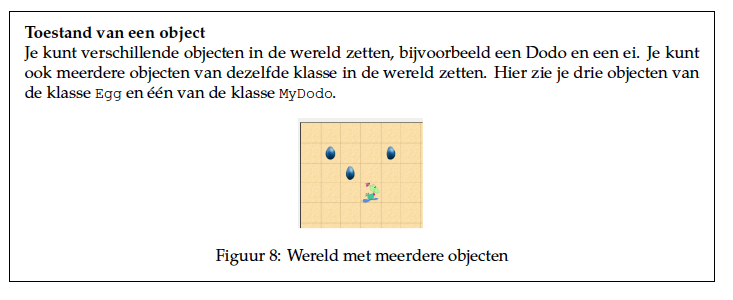




Wat zijn dan die dingen die alle Dodo’s kunnen? (mutatormethodes)

1. Noem ministens drie Dodo-methoden doe Mimi erft.
2. De methode void turnLeft() behoort ot de klasse Dodo. Kan Mimi die methode uitvoeren?
3. Bekijk het klassendiagram. Welke andere ‘is-een’ relaties zie je?
4. Zet een tweede MyDodo in de wereld. Zie je verschillen in de methodes van de nieuwe MyDodo en die van Mimi?

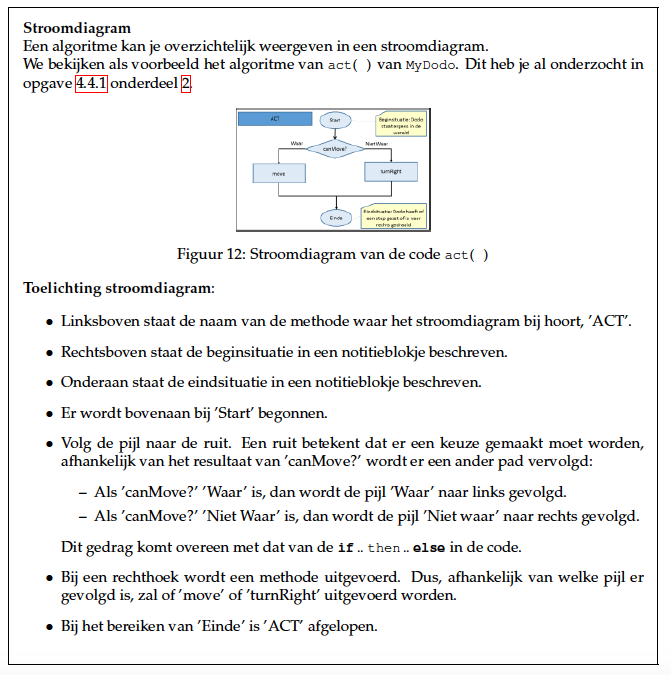
**7. Toestanden**



De toestand van een object kun je bekijken door met de rechtermuisknop op het object te klikken, en dan ‘Inspect’ te kiezen:



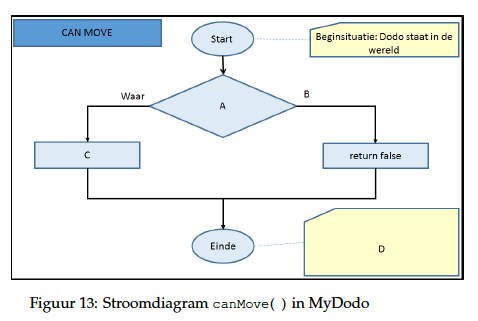
**8. Stroomdiagram**



We bekijken nu de methode canMove().

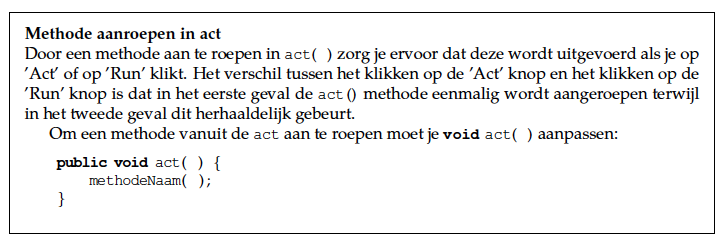
1. Open de code voor MyDodo in de editor en zoek de methode boolean canMove().
2. Leg voor elke regel code uit wat het precies betekent. (het symbool ‘!’ betekent ‘niet’).
3. Bekijk de stroomdiagram in figuur 13. Hierbij een toelichting:

* De ruit geeft een keuze aan.
* Als de conditie in ruit ‘Waar’ is dan wordt linker pijl uitgevoerd.
* Als de conditie in ruit ‘Niet Waar’ is dan wordt rechter pijl uitgevoerd.



1. Wat moet er bij A, B, C, en D in het stroomdiagram staan?

**9. Methode aanroepen in act**



1. Open de code voor MyDodo in de editor en zoek de methode void act().
2. Haal de code tussen de accolades { en } weg.
3. Roep daar de methode void turn180() aan. Kijk in het voorbeeld hierboven hoe dit moet.
4. Pas ook het commentaar boven act() aan.
5. Compileer de code.
6. Voer het programma uit met ‘Run’.
7. Test het programma met ‘Act’. Doet het programma wat je verwacht?

**Huiswerk opdrachten**

**-je kan Google gebruiken bij het maken van de antwoorden**

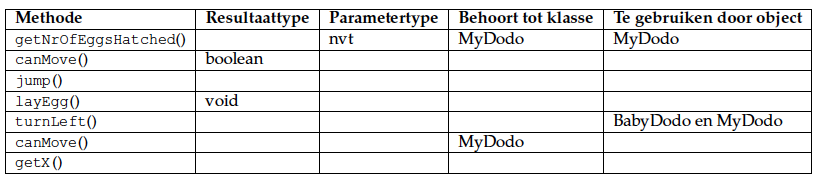
**-inleveren via je eigen Github account (voor de deadline)**

1. Leg in jouw eigen woorden uit wat een resultaattype is:

de benaming van resultaten (van uitvoering van zaken van het betreffende zaaktype) zoals de organisatie die hanteert.

1. Hieronder zie je een overzicht van de verschillende methoden. Vul deze tabel aan.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| methode | resultaattype | parametertype | behoort tot klasse | te gebruiken door object |
| getNrOFEggsHatched | int | nvt | MyDodo | myDodo |
| canMove0 | boolean |  | MyDodo | myDodo |
| jump0 | int |  | MyDodo | myDodo |
| layEgg0 | void |  | MyDodo | myDodo |
| turnLeft0 | void |  | MyDodo | BabyDodo en MyDodo |
| canMove0 | boolean |  | MyDodo | myDodo |
| getX0 | int |  | MyDodo | myDodo |



1. Wat zijn de coordinaten van de linkeronderhoek?

(1,12)

1. Hieronder zie je de code van een nieuwe methode.

Public **A** moveDown() {

turnRight();

move();

**B** move();

move();

move();

**C** move();

D turnLeft();

E move();

move();

move();

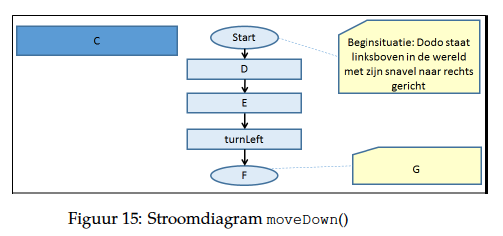
move();

F move();

G hatchegg();

}

Het bijbehorende stroomdiagram is:



Wat hoort er te staan bij **A**, **B,** C, D, E, F en G in de code en in het stroomdiagram.?

1. Smetsers-Weeda, Renske; Smetsers Sjaak. Algorithmic thinking and structured programming (in

   Greenfoot) (2015) , CC BY - NC - SA 4.0 licensed [↑](#footnote-ref-1)
2. https://creativecommons.org/licenses/by/4.0/ [↑](#footnote-ref-2)