Les 13 voorbereiding

# Theorie

## Screencast over onderwerp 10: Interfaces

<http://www.youtube.com/playlist?list=PLpd9jJvk1PjnR8YVs3ZsZtJoGbtJLSUV_>

## Boek

### Hoofdstuk 10

10.6, 10.7 en 10.9 pagina 389 t/m 398

# Opgave het woord Interface

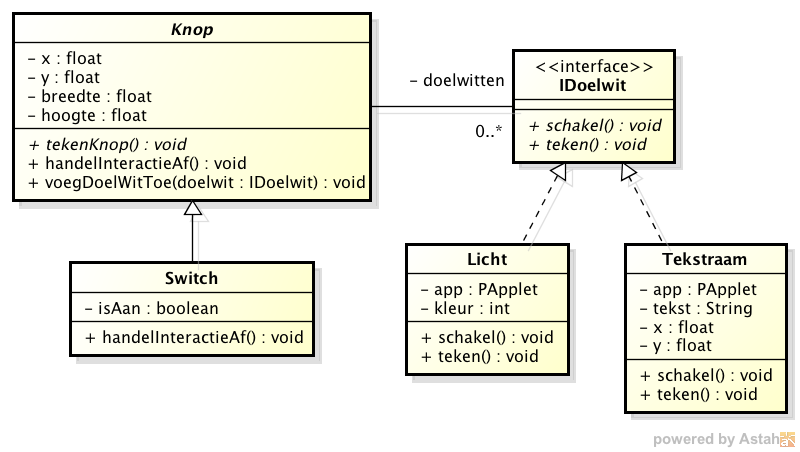
Bij het maken van (objectgeörienteerde) programma’s komt de term “interface” geregeld voorbij. Geef drie mogelijke betekenissen van het woord “interface”.

# Opgave meerdere interfaces implementeren

Zoals in de screencast wordt verteld, is het in Java niet toegestaan om van meerdere klasses te erven (“multiple inheritance” bestaat in Java niet), maar mag je wel meerdere interfaces implementeren en daarnaast zelfs nog van een klasse erven. Waarom is multiple inheritance wel een probleem, maar het implementeren van meerdere interfaces niet?

# Opgave Meerdere doelwitten

We starten met een programma dat lijkt op het programma uit de screencast, maar met een paar wijzigingen. Je vindt de startcode op Scholar en het klassediagram staat hieronder.



Een knop kan nu meerdere IDoelwitten bevatten. Zodra er op de knop geklikt wordt, moeten alle gekoppelde doelwitten de methode schakel aanroepen.

De klasse Tekstraam plaats een tekst op het scherm als de methode schakel aangeroepen wordt.

## Opdracht A

Implementeer deze klassen en laat zien dat het programma werkt door een instantie van Licht en Tekstraam aan een instantie van Switch te koppelen in het hoofdprogramma. Je kunt om te testen de gegeven code in de klasse “KnoppenApp” gebruiken: hierin wordt een switch gemaakt waaraan een instantie van licht en een instantie van doelwit worden gekoppeld.

## Opgave B

Voeg een tweede switch toe waaraan dezelfde instantie van licht is gekoppeld als aan de switch uit opgave A. Wat voor probleem kan nu optreden?

# Opgave Van if-else naar Interface

Gegeven een applicatie waarmee rechthoeken en cirkels getekend geplaatst en verwijderd kunnen worden.

import java.util.ArrayList;  
import processing.core.PApplet;  
  
@SuppressWarnings("serial")  
public class TekenApp extends PApplet {  
  
 public static void main(String[] args) {  
 PApplet.main("week5.ifelsenaarinterface.TekenApp");  
 }  
  
 private ArrayList<Figuur> figurenLijst = new ArrayList<>();  
 private String huidigGereedschap = "selecteer";  
   
 public void setup() {  
 size(400, 400);  
 }  
   
 public void draw() {  
 background(0);  
 for (Figuur figuur : figurenLijst) {  
 figuur.teken(this);  
 }  
 }  
   
 public void mousePressed() {  
 switch (huidigGereedschap) {  
 case "selecteer":  
 System.out.println("s");  
 break;  
 case "rechthoek":  
 Rechthoek r = new Rechthoek( mouseX, mouseY, 50, 50);  
 figurenLijst.add(r);  
 break;  
 case "cirkel":  
 Cirkel c = new Cirkel(mouseX, mouseY, 50);  
 figurenLijst.add(c);  
 break;  
 case "gum":  
 for (int i = figurenLijst.size() - 1; i >= 0; i--) {  
 Figuur fig = figurenLijst.get(i);  
 if (fig.isMuisBinnen(mouseX, mouseY)) {  
 figurenLijst.remove(i);  
 }  
 }  
 break;  
 }  
 }  
   
 public void keyReleased() {  
 switch (key) {  
 case 's':  
 huidigGereedschap = "selecteer";  
 break;  
 case 'r':  
 huidigGereedschap = "rechthoek";  
 break;  
 case 'c':  
 huidigGereedschap = "cirkel";  
 break;  
 case 'g':  
 huidigGereedschap = "gum";  
 break;  
 }  
 }   
}

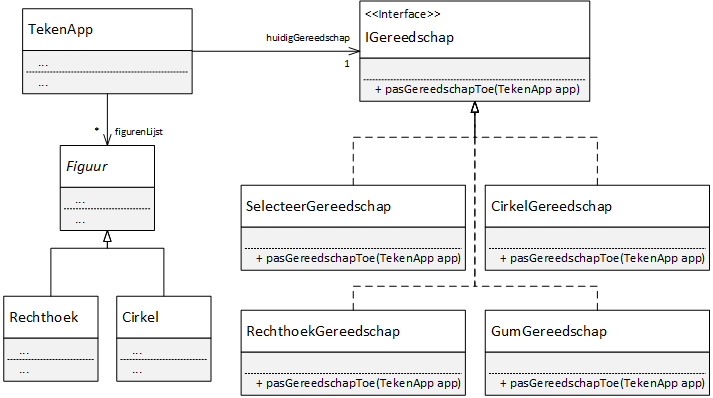
Hoewel deze code werkt, is het uitbreiden ervan lastig. Het probleem zit hem in het switch-statement in de methode mousePressed().

Zodra er een nieuw gereedschap toegevoegd moet worden, moet het gedrag van dit gereedschap toegevoegd worden op een onoverzichtelijk plek. Daarnaast moet de naam van dit gereedschap op meerdere plekken gebruikt, zonder dat de compiler kan controleren of die naam goed geschreven is.

In deze opgave ga je met behulp van een interface dit switch-statement wegwerken.

## Opdracht A

Hieronder is een klassendiagram te zien waarin een oplossing geboden wordt. Schrijf de code voor de Interface IGereedschap en voor alle gereedschappen die deze interface implementeren. Verplaatst de code uit mousePressed naar de juiste klassen.



## Opdracht B

Verander in het hoofdprogramma de code uit keyReleased() en mousPressed zodanig dat er optimaal gebruik wordt gemaakt van alle gereedschapsklasse.

Als het goed is bestaat de code in mousePressed() na aanpassing uit slechts één regel. Overigens hoef je het switch-statement uit keyReleased() alleen aan te passen. Weg werken van dit statement is niet nodig