hogeschool Windesheim

Techniek - HBO - ICT

Naam: Studentnummer: Klas:

Practicum JAVA Praktijk (108927)

Vakcode : ICT.P.JAVA2.V20 (ICT.P.JAVA.V19, ICT.P.JAVA.V18, ICT.P.JAVA.V17) (t1)
Datum : woensdag 7 april 2021
Tijd : 14.30 - 17.30 uur

Klas:	Lokaal:	Aantal:
ICTM2a t/m r , ICTM2tt	volgt	385

Opgesteld door : Wilco Moerman
Docenten : WPH01, LNR08, DSW01, DFG01, NMJ01, MNC07 VEE02, RWM02, FAP02, CNW01, SSW02, KGW01, DAS01, CSI01
Gecontroleerd door : Wietske Doornbos, Wouter Keuning, Gerben de Wolf

Rekenmachine : alle rekenmachines toegestaan
Literatuur : toegestaan
Overige hulpmiddelen : laptop

Opgaven inleveren : ja

CONTROLEER VOORAF DE VOLGENDE GEGEVENS:

Dit tentamen bevat:

5 opgaves

14 genummerde pagina's

Waarschuw de surveillant als één van deze aantallen niet klopt!

Studentnummer	Naam	Klas	Cijfer
Tijd van inleveren:			

De toets, de punten, etc.

Het gebruik van telefoons, social media, forums, dropbox en alles wat je in contact met andere personen kan brengen is tijdens de tentamentijd niet toegestaan.

Het gebruik van internet om informatie op te zoeken is wel toegestaan Je mag dus wel googlen en bv. iets lezen op een forum zoals *Stackoverflow*, maar je mag er geen vragen stellen.

In totaal zijn **100** punten te behalen. Het eindcijfer wordt verkregen door de behaalde punten te delen door **10**. Het laagst te behalen cijfer is een 1, het hoogste een 10.

Voorbereiding

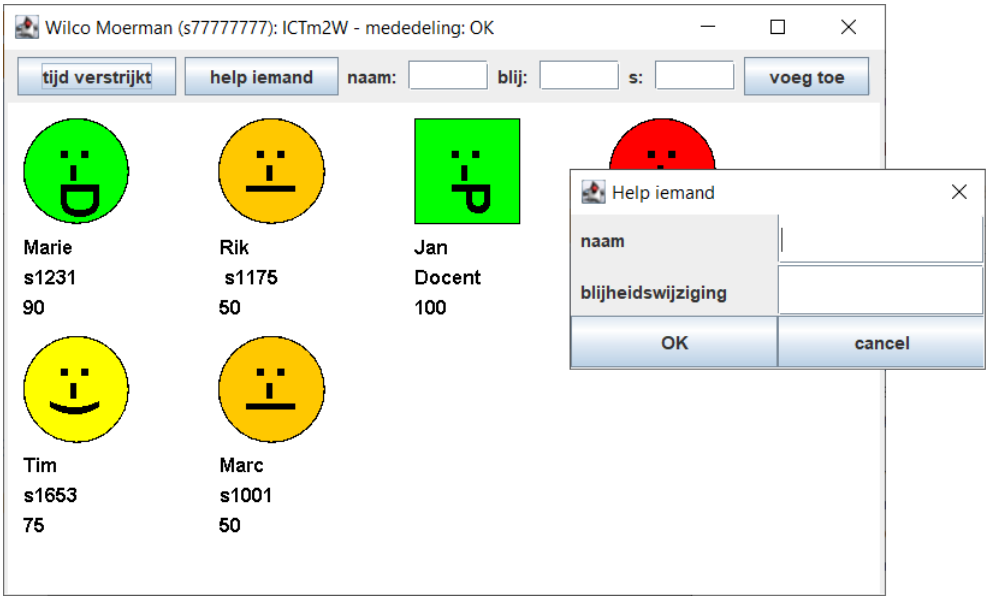
Alle code die in deze toets getoond wordt, kun je vinden in het het bestand **`"startsituatie_praktijk_7-april-2021.zip"`** in de folder `"inleverpunt praktijk 7 april"`.

De getoonde voorbeeld- en testcode is niet volledig. Je zult zelf moeten testen/onderzoeken of je code doet wat deze moet doen volgens de vraag.

De prints in de toets zijn bedoeld om de *flow* van het programma goed te kunnen volgen. Ze moeten worden toegevoegd als dat in de opgave staat, maar een spelfoutje of een spatie teveel is dus geen probleem.

Inleiding

Vanwege Covid-19 is veel onderwijs digitaal. Om digibete docenten te trainen, wil Windesheim een simulatie waarin ze kunnen oefenen om de klas tevreden te houden.



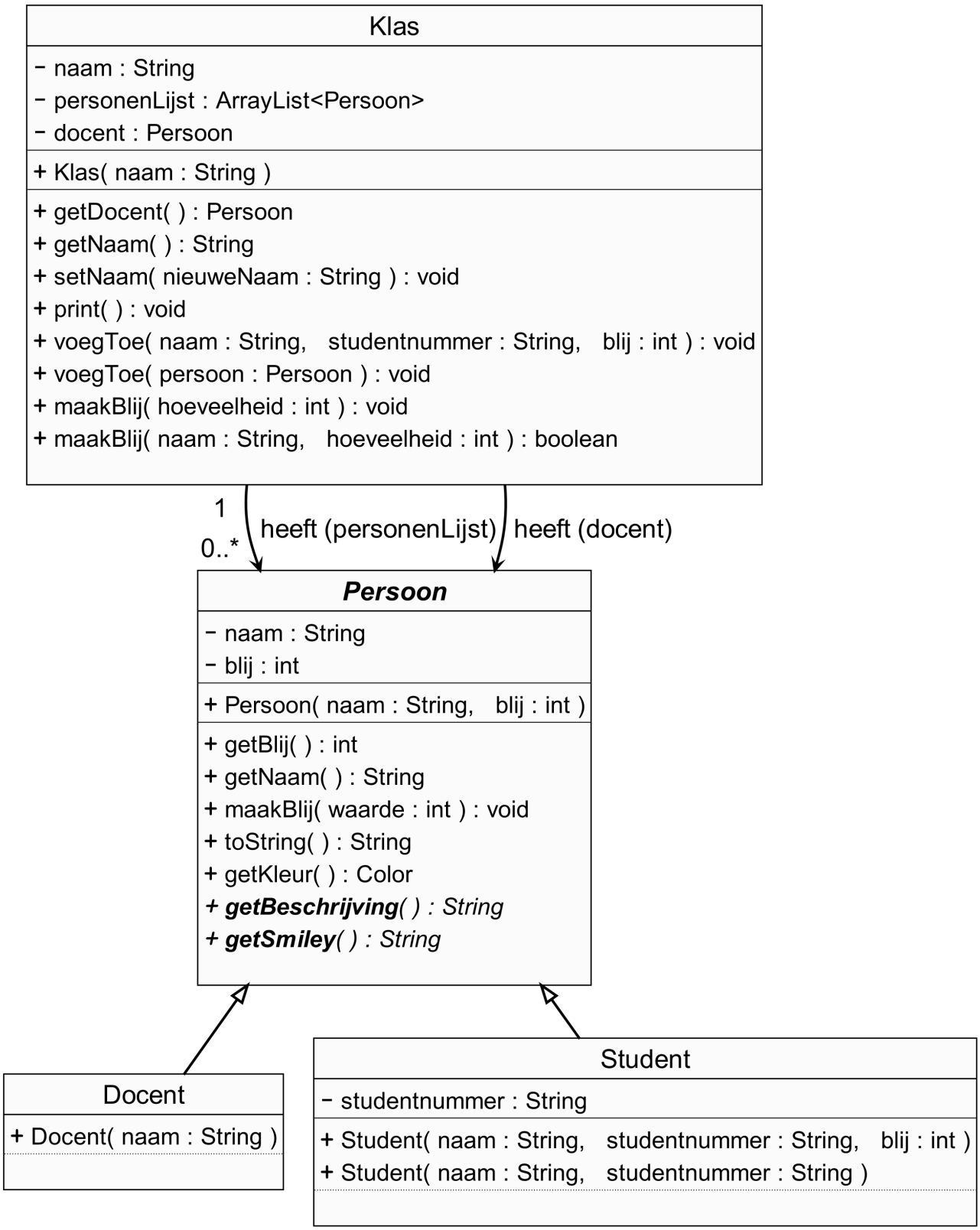
Klassendiagram

Hieronder volgt het klassendiagram voor opgave **1** en **2**. Overgeërfde methodes zijn hier **niet** in aangegeven (ook niet als je in een opgave wel gevraagd wordt ze te implementeren). Abstracte methodes zijn ***vetgedrukt en cursief***.

Constructor-parameters die dezelfde naam hebben als attributen, worden in die attributen opgeslagen.

Voor elke methode, constructor en attribuut geldt dat ze **exact** zoals in het klassendiagram aangegeven, moeten worden gemaakt. Afwijken kan puntenaftrek opleveren.

Je mag **geen** andere *methodes* of *attributen* toevoegen dan dat er in het klassendiagram staan, behalve als ze `private` zijn. Als je niet weet hoe je iets moet oplossen volgens de regels, dan kun je er vanaf wijken, zodat je verder kunt met de volgende vraag.


Klas
- naam : String
- personenLijst : ArrayList<Persoon>
- docent : Persoon
+ Klas(naam : String)
+ getDocent( ) : Persoon
+ getNaam( ) : String
+ setNaam(nieuweNaam : String) : void
+ print( ) : void
+ voegToe(naam : String, studentnummer : String, blij : int) : void
+ voegToe(persoon : Persoon) : void
+ maakBlij(hoeveelheid : int) : void
+ maakBlij(naam : String, hoeveelheid : int) : boolean
1
0..*
heeft (personenLijst)
heeft (docent)
Persoon
- naam : String
- blij : int
+ Persoon(naam : String, blij : int)
+ getBlij( ) : int
+ getNaam( ) : String
+ maakBlij(waarde : int) : void
+ toString( ) : String
+ getKleur( ) : Color
+ getBeschrijving( ) : String
+ getSmiley( ) : String
Docent
+ Docent(naam : String)
Student
- studentnummer : String
+ Student(naam : String, studentnummer : String, blij : int)
+ Student(naam : String, studentnummer : String)

Opgave 1: De personen [25 punten]

a) [15 punten]
Maak alles van de klasse `Persoon` (met uitzondering van `toString()`):

- `blij` mag **nooit** kleiner dan **0** of groter dan **100** zijn. Te kleine waardes worden op 0 gezet, te grote waardes op 100.
- De `maakBlij`-methode telt de input bij `blij` op.
- De `getKleur`-methode returnt een `java.awt.Color` op basis van `blij`.

blij is:	getKleur returnt:
kleiner of gelijk aan 25	`Color.red`
groter dan 25 maar kleiner of gelijk aan 50	`Color.orange`
groter dan 50 maar kleiner of gelijk aan 75	`Color.yellow`
groter dan 75	`Color.green`

b) [5 punten]
Maak alles van de klasse `Docent`:

- De `getBeschrijving`-methode returnt **"Docent"**.
- De `blij`-waarde van de `Docent` *begint* uiteraard altijd op **100**.
- De `getSmiley`-methode voor `Docent` returnt "**`:-P`**".
- `Docent` heeft zelf ***geen*** `toString()`.

Maak de `toString()` van de **`Persoon`**-klasse. Deze gebruikt de `naam`, `getBeschrijving()`, `blij` en `getSmiley()`. De werking staat hieronder.

toString() van Persoon / Docent. (Verwachte output staat in comments):

```
public static void main(String[] args)
{
      Docent x = new Docent( "Jan" );
      System.out.println( x );         // Jan (Docent)      100   :-P
}
```

Tip: Je mag **\t** (tab) gebruiken in een `String` om te zorgen voor een nette uitlijning.

c) [5 punten]
Maak alles van de klasse `Student`:

- Als er geen waarde voor `blij` meegegeven wordt, gebruik dan **`50`**.
- De `getBeschrijving`-methode returnt het studentnummer.
- `Student` heeft zelf ***geen*** `toString()`.

De `getSmiley`-methode werkt als volgt:

blij is:	getSmiley returnt:
kleiner of gelijk aan 25	**:-(**
groter dan 25, kleiner of gelijk aan 50	**:-|**
groter dan 50, kleiner of gelijk aan 75	**:-)**
groter dan 75	**:-D**

werking van Student-constructors. (Verwachte output in de comments):

```
public static void main(String[] args)
{
      Student st = new Student( "Anna", "s1234" );
      System.out.println( st );                    // Anna (s1234)   50    :-|

      st = new Student( "Adam", "s0012", 95 );
      System.out.println( st );                    // Adam (s0012)  95    :-D
}
```

Opgave 2: de Klas [35 punten]

a) [10 punten]
Maak de klasse `Klas` met de attributen, constructor en de methodes `getDocent`, `setNaam` en `getNaam`.

Niet elke `String` is een geldige `naam`. Voor de `naam` gelden de volgende regels:

- de `naam` is niet gelijk aan "###"
- en de lengte van `naam` is minimaal 3
- en `naam` is niet gelijk aan `null`.

Als de `setNaam` geen geldige waarde meekrijgt, wordt de `naam` niet veranderd.

Als de constructor geen geldige waarde meekrijgt, moet de `naam` `""klas???""` worden.

werking van constructor en setNaam (in de comments staat de verwachte output):

```
public static void main(String[] args)
{
      Klas klas = new Klas( "2W" );
      System.out.println( klas.getNaam() );     // klas???

      klas.setNaam( "Goede naam voor klas!" );
      System.out.println( klas.getNaam() );     // Goede naam voor klas!

      klas = new Klas( "ICTm2W" );
      System.out.println( klas.getNaam() );     // ICTm2W

      klas.setNaam( "###" );
      System.out.println( klas.getNaam() );     // ICTm2W

      klas.setNaam( "topklas" );
      System.out.println( klas.getNaam() );     // topklas
}
```

b) [15 punten]
Maak de `voegToe`- en `print`-methodes.

De `voegToe( String, String, int )`-methode voegt een nieuw `Student`-object toe aan `personenLijst` op basis van de meegegeven parameters.

De `voegToe( Persoon )`-methode voegt de input toe aan `personenLijst`.

- Als het toegevoegde object een `Docent` is, wordt deze opgeslagen in het attribuut `docent`.
- Er kan maximaal **één** `Docent` in een `personenLijst` zitten. Een tweede mag **niet** toegevoegd worden.

werking van voegToe en print (verwachte outputs in de comments):

```
public static void main(String[] args)
{
      Klas klas = new Klas( "ICTm2W" );
      klas.voegToe( "Eva", "s1234", 70 );
      Docent jan = new Docent( "Jan" );
      System.out.println( klas.getDocent() ); // null
      klas.voegToe( jan );
      System.out.println( klas.getDocent() ); // Jan (Docent)   100   :-P

      // deze toevoeging mag niet, er is al een Docent aanwezig:
      klas.voegToe( new Docent( "Josien" ) );
      System.out.println( klas.getDocent() ); // Jan (Docent)   100   :-P
      klas.print();
                                              // === Klas: ICTm2W ===
                                              // * Eva (s1234)  70     :-)
                                              // * Jan (Docent) 100    :-P
}
```

c) [10 punten]
Maak nu de `maakBlij`-methodes. De `maakBlij(int)`-methode verhoogt of verlaagt de `blij`-waardes van iedereen in de `personenLijst` met de meegegeven waarde.

De `boolean maakBlij( String, int )` verandert alleen de waarde van de `Persoon` die een `naam` heeft die gelijk is aan de meegegeven `String`.

- Als zo iemand aanwezig is, wordt zijn/haar `blij`-waarde veranderd met de meegegeven `int` en wordt `true` gereturnd.
- Als zo iemand niet aanwezig is, wordt `false` gereturnd.
- Je mag er vanuit gaan, dat er nooit meerdere personen zijn met dezelfde `naam`.

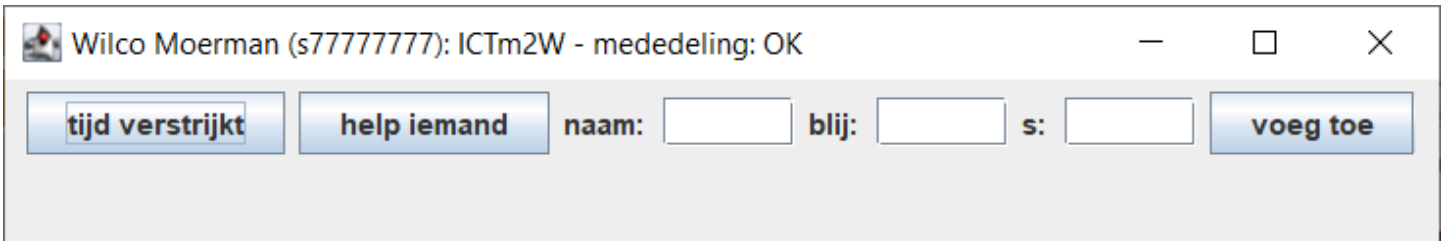
de maakBlij-methodes (verwachte output in de comments):

```
public static void main(String[] args)
{
      Klas klas = new Klas( "ICTm2W" );
      klas.voegToe( "Rik", "s1657", 80 );
      Docent jan = new Docent( "Jan" ); // begint met blij = 100.
      klas.voegToe( jan );
      klas.voegToe( "Ada", "s1131", 59 );
      klas.maakBlij( -50 );
      klas.print();
                                                  // === Klas: ICTm2W ===
                                                  // * Rik (s1657)       30    :-|
                                                  // * Jan (Docent)      50    :-P
                                                  // * Ada (s1131)       9     :-(
      //
      boolean res = klas.maakBlij( "Ada", 90 );
      System.out.println( res );                  // true
      res = klas.maakBlij( "Anoniempje", -55 );
      System.out.println( res );                  // false
      klas.print();
                                                  // === Klas: ICTm2W ===
                                                  // * Rik (s1657)       30    :-|
                                                  // * Jan (Docent)      50    :-P
                                                  // * Ada (s1131)       99    :-D
}
```

Opgave 3: GUI, het scherm [15 punten]

Vanaf deze opgave is er geen klassendiagram.
Je mag **getters** toevoegen aan klassen uit opgave 1 en 2 *voor gebruik in de GUI*.

a) [5 punten]
Maak de klasse `KlasGui` die een GUI toont zoals in de screenshot.
(Dit hoeft niet op de pixel nauwkeurig.)



Voor de `KlasGui` geldt:

- heeft een `Klas`-object als attribuut (via de constructor meegegeven).
- `FlowLayout`, ongeveer `600` bij `400` pixels.
- De applicatie wordt afgesloten door op het kruisje te klikken.
- De *titel* bevat je eigen naam en studentnummer, de `naam` van het meegegeven `Klas`-object, en de tekst `"mededeling:"`.

Voorlopig komt er de mededeling "OK" te staan.

b) [5 punten]
Maak de volgende knoppen werkend:

- de **voeg toe**-knop voegt een `student`-object toe aan de `Klas` op basis van de waardes in de tekstvelden. (Foutafhandeling komt bij (c)).
- de **tijd verstrijkt**-knop roept de `maakBlij`-methode van `Klas` aan met waarde `-5` (om te simuleren dat iedereen minder blij wordt als de tijd verstrijkt).
- Elke keer dat een knop ingedrukt wordt, moet de `Klas` geprint worden.

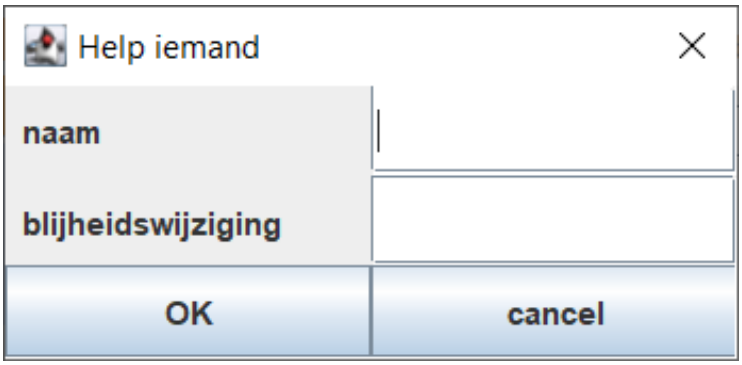
c) [5 punten]
Als je bij het tekstveld voor "blij" *geen* getal invoert, volgt een `Exception`. Los dit op met een *zo specifiek mogelijke* foutafhandeling. Er wordt in zo'n geval niemand toegevoegd.

De titel moet tonen of het goed is gegaan ("**OK**") of niet ("**Exception!**").

Opgave 4: GUI, dialoog [10 punten]

Door een `Student` te helpen, wordt deze blijer, en de `Docent` ook.
We simuleren dit door via een dialoog iemands blijheid te veranderen.

a) [5 punten]
Maak dialoog zoals in het screenshot is weergegeven.



De dialoog:

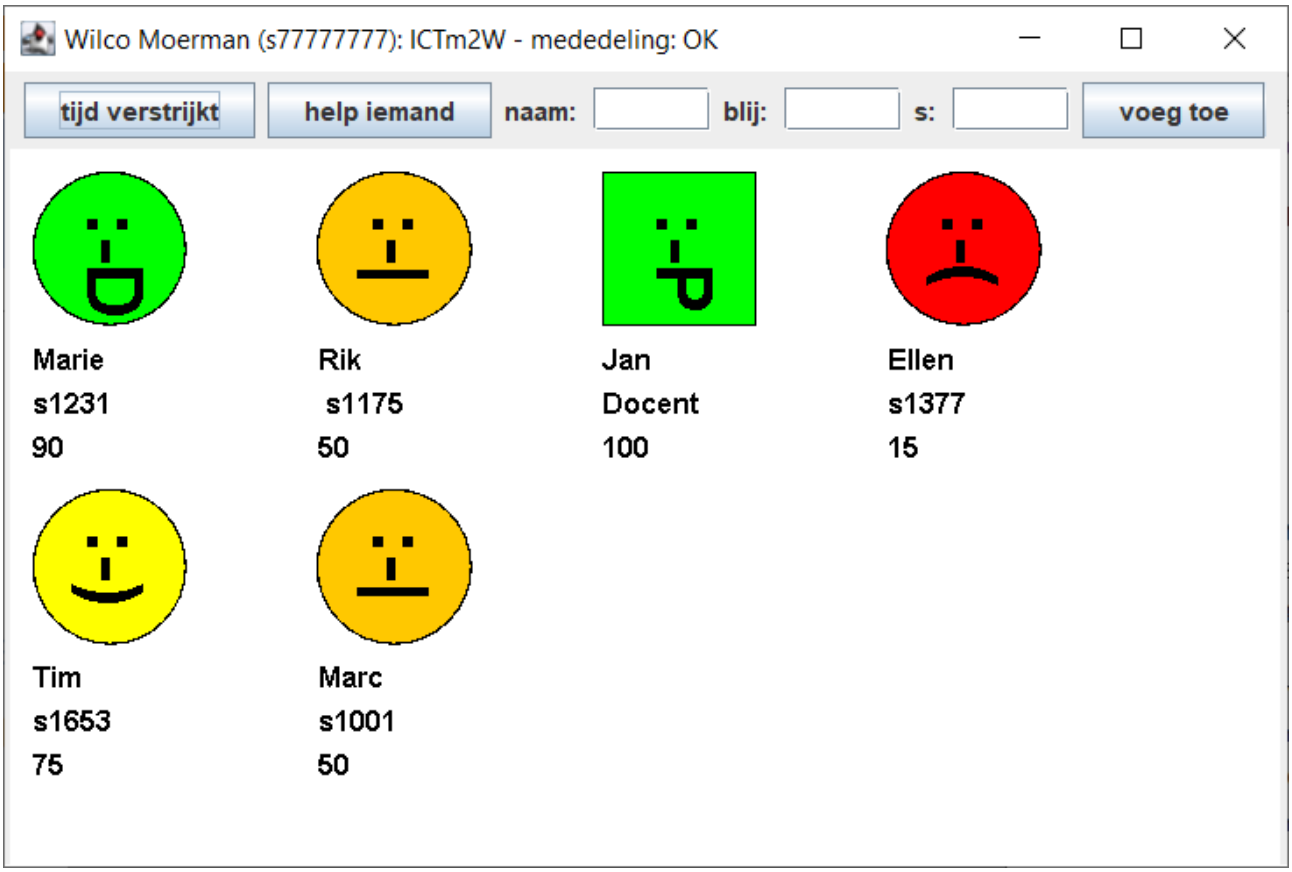
- verschijnt als de **"help iemand"**-knop wordt ingedrukt
- is modaal
- titel: "`Help iemand`"
- wordt afgesloten als op het kruisje of de knoppen geklikt wordt
- alle attributen zijn *private*

b) [5 punten]
Maak de werking als volgt:

- als de **OK**-knop gekozen is, wordt van de `Persoon` met de ingevoerde `naam` de `blij`-waarde aangepast (verhoogd of verlaagd, afhankelijk van de opgegeven waarde).
- Als er geen `Persoon` aanwezig is met de ingevoerde `naam`, dan wordt er niks gedaan.
- Als er geen `int` ingevuld wordt, gebeurt er uiteraard niks (en de GUI mag niet crashen).
- Als de blijheid van een `Persoon` ***verhoogd*** werd, dan wordt de `docent` **10** blijer, en anders niet.

Opgave 5: GUI, tekenen [15 punten]

Hieronder zie je hoe de GUI eruit moet zien:



Maak de "iconen" voor de personen als volgt:

- Een `Docent` wordt weergegeven als een vierkant van 70 x 70 pixels.
- Studenten worden weergegeven als cirkels met een diameter van 70 pixels.
- De ruimte tussen de iconen is ook ongeveer 70 pixels.
- De `getKleur`-methode van `Persoon` wordt gebruikt voor de kleur.
- Als de rij vol is, gaat het tekenen op de volgende rij verder.
- Toon onder elk icoon de gegevens (fontgrootte in het screenshot is 13).
- Teken in elke icoon het resultaat van `getSmiley` (fontgrootte 35; de tekst is 90 graden gedraaid met de methode `drawStringRotated` uit de tips hieronder).

Tips:

- Met **`setPreferredSize( new Dimension(...) )`** bepaal je de afmetingen van een `JPanel`.
- Met **`getWidth()`** kun je van je panel de huidige breedte opvragen.
- Met **`setFont`** kun je het lettertype en de grootte aanpassen. Met de code:
  `g.setFont(new Font("default", Font.BOLD, 13))`
  krijg je bv. een vetgedrukt font van 13 pixels hoog.
- Met **`drawStringRotated`** (zie onderstaande code) kun je een `String` geroteerd tekenen. Je kunt de `drawStringRotated`-methode vinden in `"startsituatie_praktijk_7-april-2021.zip"`.

code om geroteerde tekst te tekenen

```
// Plaats deze code in de klasse waar je 'm wilt gebruiken.
//
// aanroep: drawStringRotated(g2d, "hoi", 100, 200, 45);
// resultaat: tekent "hoi" op plek x=100, y=200 met een hoek van 45 graden.
//
// Inhoud van deze methode direct weer vergeten, alleen gebruiken :-)
//
// Gebaseerd op: stackoverflow.com/a/26705815
//
public void drawStringRotated( Graphics g, String txt, double x, double y,
int angle ) {
      Graphics2D g2d = (Graphics2D) g;
      g2d.translate( (float) x, (float) y );
      g2d.rotate( Math.toRadians( angle ) );
      g2d.drawString( txt, 0, 0 );
      g2d.rotate( -Math.toRadians( angle ) );
      g2d.translate( -(float) x, -(float) y );
}
```

Einde Tentamen

Maak een archief (.zip of .rar) van je **java**-bestanden. Geef het archief de volgende naam: `"java-praktijk_Voornaam_Achternaam_studentnummer.zip"` (of `...rar`).

Upload je `zip/rar`-bestand op ELO in het inleverpunt in de folder genaamd: **`"inleverpunt praktijk 7 april"`**

Als ELO je vertelt dat je het bestand niet kunt uploaden omdat er al een bestand met dezelfde naam bestaat, pas je bestandsnaam dan iets aan, bv. door er de datum aan toe te voegen).

LET OP: Kies bij het uploaden op ELO voor de **inleveren (of submit)**-knop!