hogeschool Windesheim

# Techniek – HBO-ICT

Naam: Studentnummer: Klas:

## HP JAVA Praktijk (110246)

| | |
|---|---|
| Vakcode | : ICT.P.JAVA2.V20 (ICT.P.JAVA.V19/18/17) (t1) |
| Datum | : donderdag 10 juni 2021 |
| Tijd | : 14.30 - 17.30 uur |

| Klas: | Lokaal: | Aantal: |
|---|---|---|
| ICTM2a t/m i, ICTM2n t/m r, ICTM2tt | volgt | 231,6 |

| | |
|---|---|
| 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 |
| Rekenmachine | : alle rekenmachines toegestaan |
| Literatuur | : alles: internet, boeken |
| Overige hulpmiddelen | : laptop |
| Opgaven inleveren | : ja |

CONTROLEER VOORAF DE VOLGENDE GEGEVENS:

Dit tentamen bevat:

**5** opgaves

**21** 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.

## Voorbereiding

Alle code uit deze toets kun je vinden op ELO in de folder **"inleverpunt praktijk 10 juni"** in het bestand **`"startsituatie_praktijk_10-juni-2021.zip"`**.

*De gegeven voorbeeldcode in deze toets is geen volledige test,* maar geeft een *voorbeeld* van de werking. Je zult zelf moeten testen/onderzoeken of je code alles doet wat het moet doen volgens de vraag.

De voorbeeldcode staat in de klasse `Main` per onderdeel in een losse `main(...)`-methode die je uit comments kunt halen als je 'm wilt gebruiken.

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.

Als je wilt, kun je op de papieren toets aantekeningen maken, of het klassendiagram losmaken van de rest, zodat je het naast een vraag kunt leggen.

# Inleiding

Vanwege Corona-maatregelen heeft de horeca maar beperkte capaciteit en is ze gedwongen om van elke bezoeker contactgegevens bij te houden.

In deze toets modelleer je een kroeg en de bijbehorende wachtrij die buiten staat.

Hieronder zie je een screenshot van het eindresultaat:

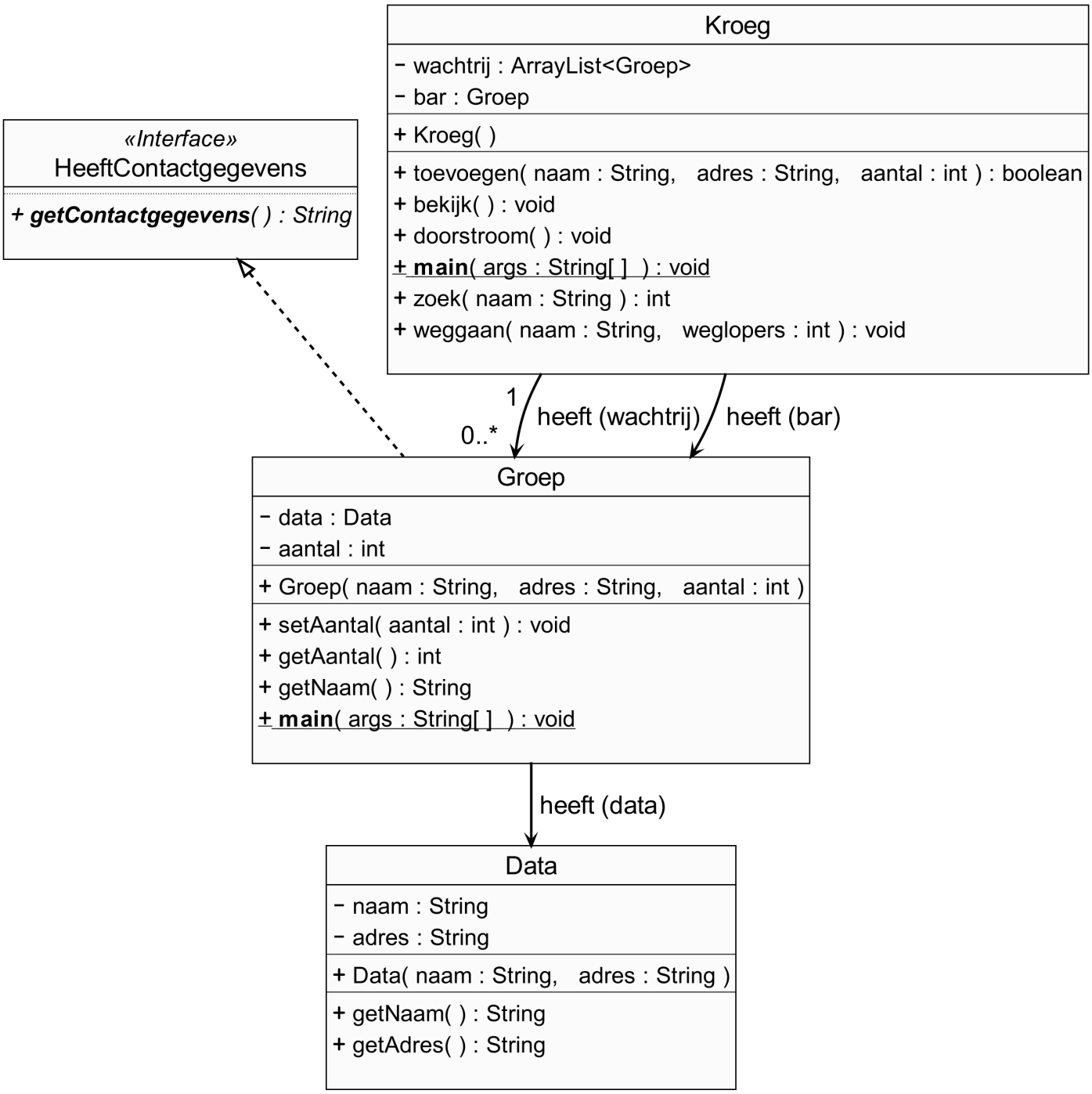

# Klassendiagram

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

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.




# Opgave 1: de wachtenden [20 punten]

**a) [10 punten]**
Maak klasse `Data` volgens het klassendiagram en zorg ervoor dat `Data`-objecten uitgeprint worden zoals in onderstaand voorbeeld.

Voor de `naam` gelden de volgende eisen:

- de lengte van de `String` is minimaal 2
- de `String` is niet gelijk aan de tekst `"??"`.
- de `String` is niet gelijk aan `null`.
- als default wordt bij ongeldige input `"onbekend"` gebruikt.

Data aanmaken en printen (verwachte output staat in comments):

```
Data d = new Data( "Wilco", "0800-JAVA" );
System.out.println( d );                    // Wilco (0800-JAVA)
Data d2 = new Data( "??", "ergens" );
System.out.println( d2 );                   // onbekend (ergens)
```

**puntenverdeling:**

**3 punten voor de basis:**

- 1 punt eraf als een attribuut niet aanwezig was en/of niet private.
- 2 aftrek als `Data` ten onrechte iets overerfde. De pijlen in het klassendiagram zijn GEEN overervingspijlen, maar gewoon 'associaties', vandaar ook dat er bij de pijl van `Groep` naar `Data` 'heeft' stond

**3 punten voor (aanwezigheid van) constructor + (werking van) methodes:**

- 2 punten eraf bij het hebben van verkeerde constructors (andere input types, afwijkend v/h diagram)
- 1 punt eraf voor elke getter die niet werkt.

**4 punten voor correcte input validatie in de constructor (attribuut naam)**

- Als te korte strings nog geaccepteerd werden: 2 eraf.
- Als `"??"` toch geaccepteerd werd (bv. omdat je niet equals maar == gebruikt om strings te vergelijken: 3 eraf.
- Als `null` als input fout ging (bv. door een crash) dan 2 punten eraf.

**2 punten voor correcte toString()**

- 1 eraf als de output een beetje afwijkt.

**b) [10 punten]**
Maak de interface `HeeftContactgegevens`

Maak alles van de klasse `Groep` overeenkomstig het klassendiagram:

- de aan de constructor meegegeven `naam` en het `adres` worden in het `Data`-attribuut genaamd `data` opgeslagen en het meegegeven `aantal` in het `aantal`-attribuut.
- de methode die aanwezig moet zijn vanwege de interface, returnt de `toString()` van `data`.
- de `getNaam`-methode returnt de `naam` die in `data` is opgeslagen.
- de `toString()` van `Groep` gebruikt de attributen `data` en `aantal`.

Groep aanmaken en printen (verwachte output staat in comments):

```
Groep g = new Groep( "Jan", "Javaweg 1", 5 );
System.out.println( g );                        // Jan (Javaweg 1): 5
System.out.println( g.getContactgegevens() ); // Jan (Javaweg 1)
```

**puntenverdeling:**

**3 punten voor alles wat met de interface te maken heeft (de interface zelf + de methode):**

- 2 punten eraf als de `getContactgegevens`-methode in `Groep` niet goed werkt.

**7 punten voor de Groep:**

- 1 punt eraf als een attribuut niet aanwezig was en/of niet private.
- 2 punten aftrek als een input niet wordt opgeslagen. Let op: de `naam` en `adres` inputs moesten in een object v/h type `Data` opgeslagen worden, niet in Strings die attribuut van `Groep` zijn

**2 punten voor de juiste toString (haken en leestekens negerend)**

- 1 aftrek als `Groep` ten onrechte iets overerfde. De pijlen in het klassendiagram zijn GEEN overervingspijlen, maar gewoon 'associaties' (heeft). Er stonden 2 pijlen van `Kroeg` naar `Groep`. De `Kroeg` heeft een ArrayList<Groep> en nog een los attribuut van type `Groep`. De pijlen kunnen sowieso geen overerving zijn, omdat je maar van 1 klasse mag overerven en er 2 uitgaande pijlen waren.

# Opgave 2: de Kroeg [40 punten]

In deze opgave maak je de `Kroeg`, inclusief `wachtrij` die buiten de `Kroeg` staat te wachten.

Als een `Groep` mensen aankomt, moeten `naam` en `adres` geregisteerd worden waarna de `Groep` toegevoegd wordt aan de `wachtrij`.

Je mag er in deze toets vanuit gaan, dat er nooit meerdere Groepen zijn met dezelfde `naam`.

De `Kroeg` heeft 1 `bar` met plek voor `5` personen.

**a) [5 punten]**
Maak de klasse `Kroeg` volgens het klassendiagram:

- maak de attributen
- je mag een constructor aanmaken, maar het is niet verplicht.

De rest van de methodes worden in de volgende onderdelen gemaakt.

**puntenverdeling:**

5 punten voor de basis van `Kroeg`.

- Daar gaan 2 punten vanaf als je geen ArrayList (`wachtrij`) hebt.
- en nog eentje als `wachtrij` niet private is.
- Verder 2 punten aftrek voor niet hebben van `bar`
- en 1 aftrek als `bar` niet private is.
- 1 punt aftrek voor gebruik van overerving/extends. `Kroeg` moest niks overerven, maar **heeft** een ArrayList met erin `Groep`-objecten en **heeft** een attribuut `bar` van type `Groep`.

**b) [15 punten]**
Maak de `toevoegen`-methode:

- als het `aantal` kleiner dan **1** of groter dan **5** is, wordt niks toegevoegd maar een bericht geprint (namelijk: `"sorry, kan niet"`)
- als de `bar` vrij (`null`) is, dan wordt de nieuwe `Groep` meteen opgeslagen in `bar` (de groep kan meteen binnenkomen)
- maar als de `bar` niet vrij is, dan wordt de nieuwe `Groep` toegevoegd aan de `wachtrij`.
- de toe te voegen `Groep` wordt uiteraard gemaakt op basis van de 3 inputs.
- als er toegevoegd is, wordt `true` gereturned, en anders `false`.

Maak ook de `bekijk`-methode. Deze print informatie over de `Kroeg`.

De werking van de `bekijk`- en `toevoegen`-methodes kun je zien in onderstaand voorbeeld.

werking van toevoegen en bekijk (verwachte outputs in de comments):

```
Kroeg k = new Kroeg();
k.bekijk();
                                                    // bar: leeg
                                                    // wachtrij: leeg
k.toevoegen( "Haantje", "de Voorste", 2 );
k.toevoegen( "Jan", "ergens", 3 );
k.toevoegen( "Anna", "thuis", 2 );
k.toevoegen( "P. Party", "0800-FEEST", 17 ); // sorry, kan niet
k.bekijk();
                                                    // bar: Haantje (de Voorste): 2
                                                    // wachtrij:
                                                    // * plek 0: Jan (ergens): 3
                                                    // * plek 1: Anna (thuis): 2
```

**puntenverdeling:**

**5 punten voor correct printen (bekijk):**

- 2 punten aftrek als de output niet correct is als de `bar` en `wachtrij` allebei leeg zijn.
- 2 punten aftrek als de output niet correct is als de `bar` bezet is maar de `wachtrij` leeg is.
- 2 punten aftrek als de output niet correct is als de `bar` bezet is en de `wachtrij` niet leeg is.
- er is niet gelet op leestekens enzo.

**10 punten voor correcte werking van toevoegen:**

- 4 aftrek als het niet goed werkt op het moment dat de `bar` leeg is.
- 4 aftrek als de `bar` niet leeg is en de `wachtrij` wel leeg, en het toevoegen niet goed gaat.
- 4 aftrek als de `bar` niet leeg is en de `wachtrij` ook niet leeg, en het toevoegen niet goed gaat.
- 1 punt aftrek als de return waarde (boolean) niet de juiste is.

**c) [5 punten]**

Op een gegeven moment moet de `Groep` aan de `bar` weg, en kan een nieuwe `Groep` aanschuiven. Dit wordt gedaan door de methode `doorstroom`.

Maak de `doorstroom`-methode:

- de eerste `Groep` wordt uit de `wachtrij` gehaald in het `bar`-attribuut opgeslagen.
- je kunt een element uit `ArrayList lijst` halen met de methode `lijst.remove(int index)`

Hieronder zie je de werking van de `doorstroom`-methode:

werking van doorstroom (verwachte outputs in de comments):

```
Kroeg k = new Kroeg();
k.toevoegen( "Haantje", "de Voorste", 2 );
k.bekijk();
                                                // bar: Haantje (de Voorste): 2
                                                // wachtrij: leeg
k.doorstroom();
k.bekijk();
                                                // bar: leeg
                                                // wachtrij: leeg
k.toevoegen( "Jan", "ergens", 3 );
k.toevoegen( "Anna", "thuis", 4 );
k.toevoegen( "Rik", "r@ik.nl", 1 );
k.bekijk();
                                                // bar: Jan (ergens): 3
                                                // wachtrij:
                                                // * plek 0: Anna (thuis): 4
                                                // * plek 1: Rik (r@ik.nl): 1
k.doorstroom();
k.bekijk();
                                                // bar: Anna (thuis): 4
                                                // wachtrij:
                                                // * plek 0: Rik (r@ik.nl): 1
```

**puntenverdeling:**

**3 punten voor de juiste inhoud van bar:**

- 2 aftrek als het 'doorstromen' niet werkt als de `wachtrij` niet leeg is
- 2 aftrek als het 'doorstromen' niet werkt als de `wachtrij` leeg is maar de `bar` bezet.
- 1 punt aftrek als het 'doorstromen' niet werkt als de `bar` leeg is.

**2 punten voor de juiste inhoud van de ArrayList wachtrij:**

- 2 aftrek als het 'doorstromen' niet werkt als de `wachtrij` niet leeg is
- 2 aftrek als het 'doorstromen' niet werkt als de `wachtrij` leeg is maar de `bar` bezet.
- 1 punt aftrek als het 'doorstromen' niet werkt als de `bar` leeg is.

**d) [15 punten]**
Op basis van de opgegeven `naam` moet de `index` (`plek`) van de `Groep` met die `naam` in de `wachtrij` bepaald worden. Maak hiervoor de `zoek`-methode:

- als er een `Groep` met de juiste `naam` aanwezig is, wordt de `plek` van die `Groep` gereturned.
- als er geen `Groep` met de juiste `naam` gevonden wordt, returnt de methode **`-1`**.

**Let op:** Als het niet lukt om de `zoek`-methode werkend te krijgen, laat deze dan `"zoek"` printen en **0** returnen, zodat je verder kunt met de `weggaan`-methode. Je kunt daar dan doen alsof de `zoek`-methode de juiste index gevonden heeft.

Zo nu en dan verandert de `wachtrij` omdat er mensen geen zin meer hebben om in de rij te staan en weggaan.

Maak de `weggaan`-methode:

- deze heeft als inputs de `naam` van de gezochte `Groep` en het aantal `weglopers`.
- als er een `Groep` met de juiste `naam` aanwezig is en `weglopers` groter dan 0 is, dan wordt het `aantal` aangepast
- als het nieuwe `aantal` kleiner dan **1** is, moet de `Groep` uit de `wachtrij` verwijderd worden.
- in alle andere gevallen wordt niks gedaan.

werking van weggaan (verwachte outputs in de comments):

```
Kroeg k = new Kroeg();
k.toevoegen( "Haantje", "de Voorste", 2 );
k.toevoegen( "Anna", "thuis", 4 );
k.toevoegen( "Jan", "ergens", 3 );
k.toevoegen( "Rik", "r@ik.nl", 3 );
k.toevoegen( "L. Aatste", "achteraf", 1 );
k.bekijk();
                                                // bar: Haantje (de Voorste): 2
                                                // wachtrij:
                                                // * plek 0: Anna (thuis): 4
                                                // * plek 1: Jan (ergens): 3
                                                // * plek 2: Rik (r@ik.nl): 3
                                                // * plek 3: L. Aatste (achteraf): 1

System.out.println( k.zoek( "Anna" ) );     // 0
k.weggaan( "Anna", 1 );
k.weggaan( "Jan", 100 );
k.weggaan( "Rik", -20 );
k.bekijk();
                                                // bar: Haantje (de Voorste): 2
                                                // wachtrij:
                                                // * plek 0: Anna (thuis): 3
                                                // * plek 1: Rik (r@ik.nl): 3
                                                // * plek 2: L. Aatste (achteraf): 1
```

**puntenverdeling:**

**8 punten voor de werking van zoek:**

- 5 punten aftrek als de methode niet goed werkt als er geen `Data` met de te zoeken `naam` in de `wachtrij` zit.
- 5 punten aftrek als je Strings vergeleken hebt met == in plaats van `equals`
- 2 punten aftrek als je Strings vergeleken hebt met `contains` in plaats van equals. Het gaat dan mis als je een langere en kortere tekst met overlap hebt: `"Jantien".contains("Jan")` is `true`.

**7 punten voor de werking van weggaan:**

- 3 punten aftrek als het fout gaat met negatieve weglopers (dus: het aantal weglopers is negatief). Als dan het aantal in de `Groep` verandert,  is dat fout.
- 3 punten aftrek voor fouten als het aantal weglopers groter is dan het aantal in de `Groep`. In dat geval zou je natuurlijk op 0 uit moeten komen,

  niet op een negatief aantal.

# 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*.

Als je een methode uit opgave 1 of 2 nodig hebt die niet werkt, zorg er dan voor dat die methode wel compileert en de naam van de methode uitprint, zodat je 'm in de GUI kunt gebruiken en ook ziet dat er wat gebeurt.

**a) [5 punten]**
Maak een GUI zoals in onderstaand screenshot:


Voor de GUI geldt:

- heeft een `Kroeg`-object als attribuut dat in de constructor aangemaakt wordt.
- `FlowLayout`, ongeveer `650` bij `350` pixels.
- de applicatie wordt afgesloten door op het kruisje te klikken.
- de *titel* bevat je eigen naam en studentnummer
- de `"toevoegen"`-knop is in het begin groen.
- je kunt een `knop` een kleur geven met: `knop.setBackground(Color.green)`

**puntenverdeling:**

- 1 punt voor het correct aanmaken van een nieuw `Kroeg`-object in de constructor van de GUI. (dus 0 punten als je een `Kroeg`-object meegaf als parameter aan de GUI).
- 1 punt voor een werkend JFrame/window
- 1 punt voor correcte layout
- 1 punt voor correcte titel
- 1 punt voor `EXIT_ON_CLOSE`

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

- als op de **"toevoegen"**-knop geklikt wordt, moet een `Groep`-object toegevoegd worden aan het `Kroeg`-attribuut m.b.v. de `toevoegen`-methode, op basis van de waardes in de tekstvelden (foutafhandeling komt bij **3(c)**).
- als op de **"doorstroom"**-knop geklikt wordt, moet de `doorstroom`-methode van het `Kroeg`-attribuut uitgevoerd worden.

Elke keer dat op 1 van de 3 knoppen geklikt wordt, moet het `Kroeg`-attribuut geprint worden met de `bekijk`-methode.

De kleur van de **"toevoegen"**-knop geeft de status van het toevoegen aan:

- als het toevoegen van een `Groep` geslaagd is, dan wordt deze knop **groen**
- als het niet geslaagd is (`aantal` te groot of te klein), dan wordt deze knop **oranje**

**puntenverdeling:**

- 1 punt voor het succesvol aanroepen van de `doorstroom`-methode als de **"doorstroom"**-knop gekozen wordt
- 1 punt voor het succesvol aanroepen van de `toevoegen`-methode als de **"toevoegen"**-knop gekozen wordt (voor beide knop-acties is nodig: alles m.b.t. actionListener etc.. Maar het is niet erg als de methode zelf niks doet. Dat is bij de vorige opgave aan de orde gekomen).
- 2 punten voor de goede werking van de groene/oranje kleur van de **"toevoegen"**-knop.
- 1 punt voor het aanroepen van de `bekijk`-methode na **elke** button-klik.

**c) [5 punten]**
Als je bij het tekstveld voor "aantal" *geen* getal invoert en toch wilt toevoegen, volgt een bepaald soort `Exception`:

- los dit op met een *zo specifiek mogelijke* `try ... catch`.
- er wordt in dit geval geen `Groep` toegevoegd aan het `Kroeg`-attribuut.
- de **"toevoegen"**-knop wordt in dit geval **rood**.

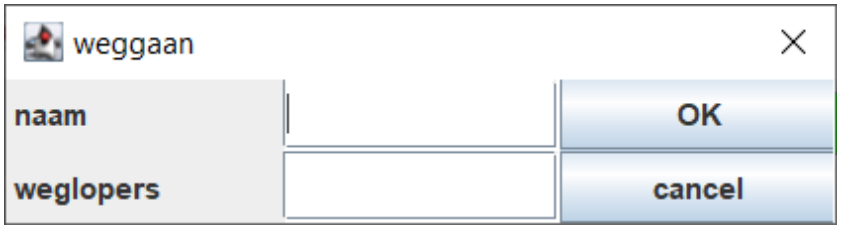
**puntenverdeling:**

- 3 punten voor de correcte Exception (moest **zo specifiek mogelijk**, dus: `NumberFormatException`)
- 2 punten voor het upaten van de **"toevoegen"**-knop naar rood als er een fout op was getreden.

# Opgave 4: GUI, dialoog [10 punten]

Soms gaan mensen uit de wachtrij weer weg. Via een dialoog ga je de `wachtrij` van het `Kroeg`-attribuut van de GUI veranderen.

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




De dialoog verschijnt als de **"weggaan"**-knop wordt ingedrukt:

- is modaal en heeft als titel `"weggaan"`
- wordt afgesloten als op het kruisje of de knoppen geklikt wordt
- alle attributen zijn *private*

**puntenverdeling:**

- 1 punt voor het succesvol laten verschijnen van de dialoog als op de **"weggaan"**-knop geklikt wordt
- 1 voor het modaal zijn.
- 1 punt voor titel en layout samen ok.

- 1 punt voor het correct afsluiten van de dialoog bij elke knop (n.b. als je een dialoog had die bij "ok" open bleef staan als de input geen getal was, dan was dat ook goed. Idem voor varianten waarbij een foutmelding werd getoond in zo'n geval).
- 1 punt voor "alle attributen private"

**b) [5 punten]**
Als er op de **"OK"**-knop geklikt wordt:

- dan wordt de wijziging doorgevoerd m.b.v. de `weggaan`-methode.
- als er geen `int` ingevuld wordt bij `aantal`, gebeurt er uiteraard niks
- de GUI/dialoog mag niet crashen.

**puntenverdeling:**

- 2 punten voor de werking bij "ok", dus succesvol uitvoeren van de `weggaan`-methode (of die methode zelf nu werkte of niet).
- 1 punt voor de correcte "canel" (dus: geen wijzigingen uitvoeren)
- 2 punten voor het niet optreden van exceptions (bv. mbv. try-catch...)

# Opgave 5: GUI, tekenen [15 punten]


Hierboven zie je een voorbeeld van het eindresultaat van de GUI, nadat er een paar Groepen toegevoegd zijn met de "`toevoegen`"-knop.

Hieronder staat wat getekend moet worden (dit hoeft niet op de pixel nauwkeurig):

- de linkerkant van het scherm is de `Kroeg`. Binnen in de `Kroeg` het licht (wit), maar de rest van het scherm is zwart, dat is 'buiten' waar het donker is.
- zet de woorden `Kroeg` en `wachtrij` op de juiste plek neer.
- als er een `Groep` aan de `bar` in de `Kroeg` zit, dan worden er evenveel groene cirkels getekend, als het `aantal` van de `Groep`.
- als de `bar` leeg is, wordt `"leeg hier!"` getekend in het `Kroeg`-gedeelte.
- elke `Groep` in de `wachtrij` wordt weergegeven met rode cirkels (evenveel als het `aantal` van die `Kroeg`)
- de cirkels hebben een diameter van 30 pixels.
- boven elke `Groep` wordt het resultaat van de `getContactgegevens`-methode getekend. (let op: `g.setColor(...)` werkt ook voor tekst)

**puntenverdeling:**

- 1 punt voor het hebben van een werkend tekenpanel (ook al werd er niks getekend)

- 3 punt voor het hebben van wit aan linkerkant en zwart aan rechterkant (en overig kleurgebruik)
- 2 punten voor het correct weergeven van de `bar` (dus: een `Groep` tekenen op de juiste plek als de `bar` niet leeg is, en anders niks tekenen daar.
- 3 punten voor het correct weergeven van de `wachtrij` (maar 1 punt aftrek als de cirkels zo dicht bij elkaar staan dat ze de corona-maatregelen overtreden)
- 2 punten voor de juiste teksten bij elke `Groep` (moeten uiteraard zichtbaar zijn, dus zwarte tekst op zwarte achtergrond levert geen punten op)
- 1 punt als er "leeg hier" (of een variant daarop) getoond werd in de `Kroeg` (het witte gedeelte) als de `bar` leeg (null) was.
- 1 punt voor correct aanroepen van super.paintComponent(g) (en niet super.paintComponent**s**(g) - dat is wat anders namelijk en werkt niet)
- 2 punten voor het aanroepen van repaint() in de `actionPerformed`-methode in de GUI, zodat de GUI altijd actueel/up-to-date is en toevoegen/wijzigen/doorstromen dus zichtbaar effect heeft.

Het maakt niet uit hoe/waar je het tekenen regelt. Je kunt alles in het tekenPanel regelen, of in de GUI, of in de `Kroeg`, of een combinatie ervan. Het enige dat telt, is dat de juiste dingen getekend worden.

**Tips:**

- met `setPreferredSize(new Dimension(...))` stel je de afmetingen van een `JPanel` in.
- met `g.setFont(new Font("default", Font.PLAIN, 14))` kun je het font wat groter/leesbaarder maken.

# 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 10 juni"`**

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