

**Object Georiënteerd Ontwerp**

Zelfevaluatie Instrument

Van Rompaey en Raf

2017-2018, Toegepaste Informatica en Reeks

18-8-2018

git-repo: https://github.com/RafVanRompaey-r0663398/Project-Herkansing\_R0663398.git  
  
University Colleges Leuven-Limburg  
Campus Proximus  
Geldenaaksebaan 335  
3001 Heverlee  
Tel. +32 (0) 16 375 700

Inhoud

[Algemene opmerking 3](#_gjdgxs)

[Vereisten 3](#_30j0zll)

[Design patterns 5](#_3znysh7)

[Speciale topics 7](#_2et92p0)

[Werkverdeling 8](#_3dy6vkm)

[Klassediagramma 9](#_tyjcwt)

# Algemene opmerking

Dit verslag gaat over de opdracht ‘zelfevaluatie instrument’, dat zal meetellen voor 5 punten van je totaalscore van dit vak. Dit verslag vormt de toelichting bij jouw code.

**Je wordt geacht om je voor het verslag volledig aan deze template te houden. Buiten deze ‘algemene opmerking’ zijn alle paragrafen verplichte onderdelen van het verslag.**

**EXTRA: Je bent verplicht om je in code gebruik te maken van het @author commentaarveld!**

# Vereisten

Geef voor het gehele project aan welke vereisten (evt. zelf verder uitgewerkt/opgesplitst aan de hand van de opgave) je succesvol hebben geïmplementeerd, en welke topics niet gelukt zijn. Indien je een deel van de voorziene vereisten niet afgewerkt hebt, geef dan aan waarom niet… De reden kan “tijdsgebrek” zijn, het kan een issue zijn “wist niet hoe, het crashte”, of het kan zijn dat je een zeer goede reden had om het niet te implementeren…

Voeg het definitieve gegenereerde klassendiagramma van je code toe, als afzonderlijke afbeelding, als bijlage bij dit rapport.

Als er zaken zijn uit de opdracht die je niet hebt kunnen uitwerken of die je beter zou willen uitwerken, dan som je deze hier tevens op (in laatste rij van tabel)

|  |  |  |
| --- | --- | --- |
| User story | OK? | indien niet ok – wat lukt niet (zie acceptatie-criteria) en waarom? |
| 1. Overzicht categorieën tonen | JA |  |
| 1. Nieuwe categorie toevoegen | JA |  |
| 1. Overzicht vragen tonen | JA |  |
| 1. Nieuwe vraag toevoegen | JA | technische info: de statement die in het text veld staat boven de text area is het correcte antwoord: |
| 1. Test afleggen | JA |  |
| 1. Resultaat tonen met punten | JA |  |
| 1. Resultaat tonen met feedback | JA |  |
| 1. Type feedback kiezen | JA/NEE | je kan het kiezen maar de property file zit in de executable jar dus je kan er niet aan |
| 1. Resultaat tonen zonder fouten | JA |  |
| 1. Reeds afgelegd | JA |  |
| 1. Test opnieuw afleggen | NEE | er is een bug dat de test niet helemaal opnieuw wordt gedaan, en daar de eerste vraag skipt. |
| 1. Bestaande categorie aanpassen | NEE | tijdsgebrek en vond niet op google hoe ik een rij uit de tableview moest hallen met de clicked on mouse event. |
| 1. Bestaande vraag aanpassen | NEE | tijdsgebrek en idem als story 12 |
| 1. Vraag toekennen aan meerdere categorieën | NEE | tijdsgebrek |
| Andere … | NEE |  |

# Design patterns

Geef voor elk gezien patroon aan waar je het gebruikt hebt (mogelijkerwijs meer dan eens). Genereer een klassendiagram voor elke situatie waarin je het patroon gebruikt hebt.

Geef extra informatie (voordelen / waarom / …). Als je een patroon niet toegepast hebt, leg je uit waarom niet.

|  |  |  |
| --- | --- | --- |
|  | Toegepast (ja/nee)  In welke stories(nr)  Waarom toegepast(voordeel) | Bijhorend klassendiagram (uit object aid) |
| Observer | elke handler gebruikt het observer patroon (ze luisteren of er iets veranderd)   * Het ondersteunt het principe van loose coupling * observers kunnen op elk moment worden toegevoegd / verwijderd. * gegevens kunnen makkelijk verstuurd worden tussen subject en observer |  |
| Strategy | niet gebruikt |  |
| Simple Factory | gebruikt bij story 4 en story 5   * isoleert de concrete klassen JaNeeVraag.java en MulitpleChoiceVraag.java * en we beheren de hoeveelheid objecten dat er gemaak worden * polymorphism |  |
| Façade | gebruikt bij story 1, story 3 en story 5   * dit verbergt voor de controller de complexiteit van de achterliggende databanken en evaluatie klasse |  |
| Singleton | niet gebruikt |  |
| State | story 5,story 6,story 9 en story 10   * dit zorgt er voor dat we een hoop code uit sparen anders zouden we veel if-statements gebruiken of een switch |  |
| MVC | heel het project gebruikt MVC | alle klassen zijn in het project opgedeeld in drie packages model, view en controller in de model package zitten alleen maar klassen die betrekking hebben tot de informatie van de programma en de berekeningen ,in de view package zitten alleen maar klassen die betrekking hebben tot het tonen van de doorgeven informatie en input vragen van de gebruiker en in de controller package zitten alleen maar klassen die betrekking hebben tot de informatie van de model aan de view geven of het ophalen van de informatie voor de view.  dit alles is zichtbaar in het klassendiagram. |
| Decorator | niet gebruikt |  |

# Speciale topics

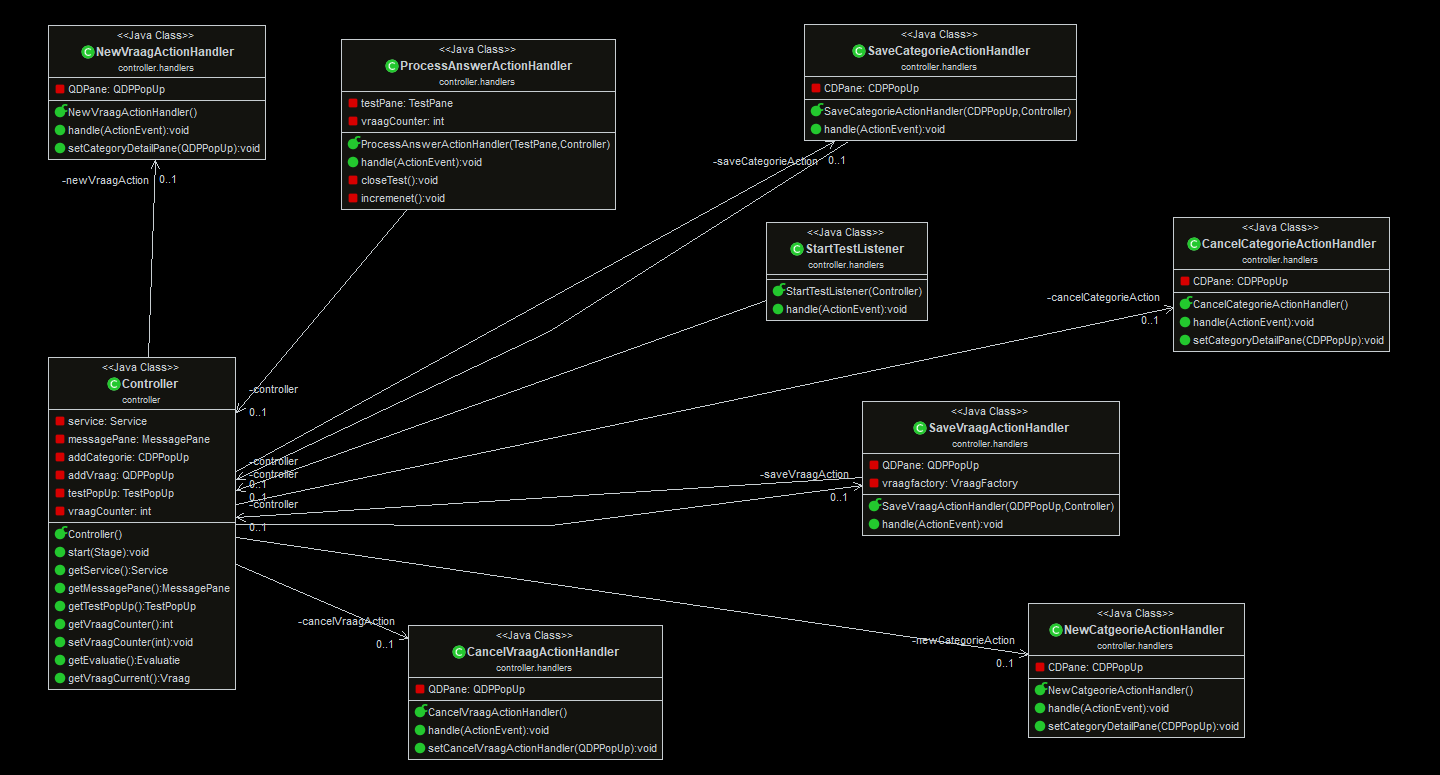
Geef voor elke “speciale topic” aan of je het gebruikt hebt of niet, en zo ja, waar. Toon eventueel aan aan met een klassendiagramma.

|  |  |  |
| --- | --- | --- |
|  | Toegepast (ja/nee)  In welke stories(nr)  Waarom toegepast(voordeel) | Bijhorend klassendiagram (uit object aid) of extra uitleg |
| Enum | nee |  |
| Properties | story 8 |  |
| Reflection | nee |  |
| Andere… | nee |  |

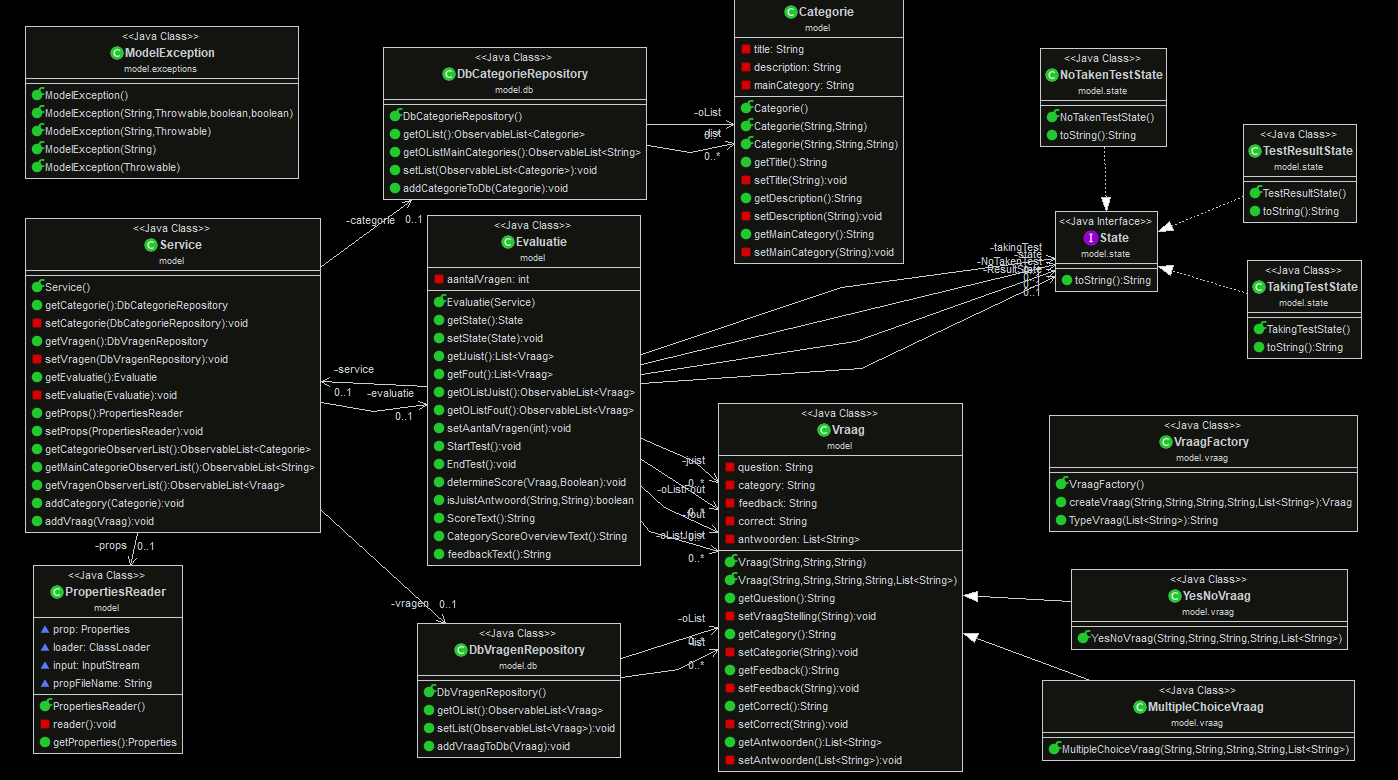
# Klassediagramma

Voeg het definitieve gegenereerde klassendiagram van je code toe (object aid), als afzonderlijk afbeelding-bestand. **Dit klassendiagram moet goed leesbaar zijn** -> Spreiden over meerdere pagina”s (vb 1 blad per package (MVC) en 1 overzicht klassendiagram (zonder attributen en methoden)

Controller:



Model:



View:

