

**Object Georiënteerd Ontwerp**

Zeeslag

[Daan Adams, Thomas Goris, Surendra Sapkota, Kelvin Schoofs]

[2016-2017, Toegepaste Informatica 2TX-2]

[25/12/2016]

University Colleges Leuven-Limburg  
Campus Proximus  
Geldenaaksebaan 335  
3001 Heverlee  
Tel. +32 (0) 16 375 700

Inhoud

[Algemene opmerking: 3](#_Toc403305641)

[Vereisten 3](#_Toc403305642)

[Patterns 5](#_Toc403305643)

[Speciale topics 6](#_Toc403305644)

[Werkverdeling 7](#_Toc403305645)

[Klassediagramma 8](#_Toc403305646)

# Algemene opmerking:

Dit verslag gaat over de opdracht ‘Zeeslag’, dat zal meetellen voor 5 punten van je totaalscore van dit vak. Je dient je code in via GIT voor elk nieuw labo. Dit verslag vormt de toelichting bij jouw code. Binnen dit document gebruiken wij een Chronometer project als voorbeeld bij de verschillende topics. Jij mag natuurlijk enkel voorbeelden uit je eigen project aanhalen.

Je wordt geacht om je voor het verslag volledig aan deze template te houden. Buiten deze ‘algemene opmerking’ zijn alle hoofdingen verplichte onderdelen van het verslag. Je dient dit verslag in via Toledo voor de allerlaatste les van dit vak.

**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. Zie hieronder voor een voorbeeld van een hele andere toepassing, met name Chronometer. 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 op.

Een voorbeeldje van mogelijke vereisten voor je project kan je hieronder terugvinden. Dit is **zeker niet** volledig! Vul het verder aan naargelang de vordering van het project. Zorg dat je niets vergeet!

Zeeslag

|  |  |  |
| --- | --- | --- |
| Vereiste | OK? | Issues? (indien niet ok) |
| Bij het starten wordt *App zeeslag getoond.* | ja | / |
| De zeeslag App heeft 2 borden: 1 voor de spelers en 1 voor de computer (AI). | ja | / |
| Zeeslag heeft een apart shippicker panel, waarmee je via de afrollijst kan kiezen welke boot je in het board wil plaatsen. | ja | / |
| Wanneer een speler een boot plaatst wordt gecontroleerd of dit een geldige locatie is. | ja | / |
| Een speler kan 5 boten plaatsen op het bord zolang het spel niet begonnen is. | ja | / |
| De computer (AI) plaatst schepen op het bord. | ja | / |
| De speler kan met een checkbox kiezen of hij zijn boot horizontaal of verticaal wilt plaatsen. | ja | / |
| Er zijn ook 2 knoppen voorzien: ‘Start’ en ‘Reset’ | ja | / |
| Zo lang men geen 5 boten plaatst op het bord kan de startknop niet worden aangeklikt. | ja | / |
| Wanneer de gebruiker op de startknop klikt, wordt deze knop onbeschikbaar gemaakt. | ja | / |
| De speler kan op het bord klikken om te schieten. | ja | / |
| Wanneer er op een schip wordt geschoten, is dit zichtbaar op het bord. | ja | / |
| De computer schiet tijdens zijn beurt op het bord van de speler. | ja | / |
| De score van de spelers wordt getoond. Tijdens het spel wordt dit ook aangepast | ja | / |
| Wanneer alle boten zijn gezonken wordt het spel beëindigd. | ja | / |

7

|  |  |  |
| --- | --- | --- |
| Vereiste | OK? | Issues? (indien niet ok) |
| Wanneer de gebruiker op de reset knop klikt wordt het bord volledig leeggemaakt. (technisch implementatie: nieuwe boarden gemaakt createcell(x,y)). | ja | / |
| Wanneer de gebruiker op maximize klik word het venster groter maar de grootte van de borden wordt niet gewijzigd. | ja | / |
| Als de gebruiker op X klikt, sluit de applicatie. | ja | / |

# Patterns

Geef voor elk gezien patroon aan waar je het gebruikt hebt (mogelijkerwijs meer dan eens). Genereer een klassendiagramma voor elke situatie waarin je het patroon gebruikt hebt. Ons gegeven voorbeeldje komt uit het Chronometer project. Jij gebruikt uiteraard alleen voorbeelden uit je eigen project.

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

|  |  |  |
| --- | --- | --- |
| Observer | Ja  In BattleshipBoardFrame:  BattleshipGame luistert naar het drukken van de Start en Reset knop;  In BattleshipBoard:  UserInterfaceController luistert naar het updaten van een cell op het board |  |
| Strategy | Ja, Strategy: doTurn en placeBoats verschilt per strategy |  |
| Façade | Ja: BattleshipGame functioneert als facade voor de andere model klassen. UserInputController functioneert als facade voor een aantal klassen, zoals de ShipPickerPanel |  |
| Builder  +  Singleton | In Strategy maken we gebruik van een Builder “StrategyBuilder” die door alle strategiën als een singleton wordt aangemaakt |  |
| State | Ja: BattleshipCell heeft een staat BattleshipCellState voor combinaties wel/niet geschoten/boot aanwezig. BattleshipGame heeft een TurnState voor wie aan de beurt is of dat het spel startend/geëindigd is |  |
| Template | Ja: Strategy (doTurn en placeBoats) |  |
| MVC | ja | Zie UML overzicht |

Je mag zoveel rijen toevoegen als je nodig hebt!

# Speciale topics

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

|  |  |  |
| --- | --- | --- |
| Enum | Ja: boat |  |
| Properties | Ja: overal |  |
| Class | Ja: overal |  |
| JDBC | Nee (geen data om op te slaan) |  |
| Mockito | Ja: test klassen |  |
| Lambda-expressies | Ja: voor alle functionele interfaces |  |

# Werkverdeling

Geef aan in percentages hoeveel je bij benadering gespendeerd hebt aan deze opdracht, **buiten de contacturen**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Thomas | Kelvin | Surendra | Daan |
| Ontwerp | 30 | 20 | 25 | 25 |
| Klassendiagrammen | 30 | 20 | 30 | 20 |
| Implementatie | 20 | 35 | 25 | 20 |
| Verslag | 30 | 20 | 30 | 20 |

# Klassediagramma

Voeg het definitieve gegenereerde klassendiagram van je code toe, als afzonderlijk afbeelding-bestand.

