

**Object Georiënteerd Ontwerp**

Zeeslag

Faouzi Mechraoui

2de jaar TI, reeks 2

22/12/2016

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.

Chronometer (voorbeeld)

|  |  |  |
| --- | --- | --- |
| Vereiste | OK? | Issues? (indien niet ok) |
| Bij het starten wordt *ChronometerFrame* getoond | ja | / |
| De *ChronometerFrame* heft een *InfoField* met een *timeTextField* | ja | / |
| De *ChronometerFrame* heft een ControlPanel met 3 knoppen | ja | / |
| Wanneer men op de start knop klikt, begint de tijd te lopen en *timeTextField* wordt geupdate. | ja | / |
| Wanneer men op de start knop klikt, wordt deze gedisabled | nee | Niet geïmplementeerd, geen tijd |
| … |  |  |
| Wanneer er op de ‘stop’ knopt wordt gedrukt, wordt deze gedisabled. | no | Een tweede keer op ‘stop’ drukken is geen issue, dus geen disabling nodig. |

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!

|  |  |  |
| --- | --- | --- |
| Vereiste | OK? | Issues? (indien niet ok) |
| Bij het starten verschijnt er een venster waar het spel op word gespeeld | OK | / |
| Elke speler heeft een eigen spelbord. | OK | / |
| Het spelbord is 10 op 10 groot (zie mockup) | OK | / |
| Het bord van de speler en van de tegenspeler (de computer) worden getoond. | OK | / |
| De naam van de speler wordt gevraagd en bij het bord getoond | OK | / |
| Als naam van de tegenspeler word “Computer” getoond | OK | / |
| De speler kan zijn schepen op het bord horizontaal of verticaal plaatsen | OK | / |
| De speler kan maar 5 schepen plaatsen | OK | / |
| De speler kan maar een bepaald aantal schepen van elke soort plaatsen | OK | / |
| De schepen kunnen niet tegen elkaar liggen en kunnen elkaar niet kruisen | OK | / |
| De AI plaatst zijn schepen volgens de regels | OK | / |
| De speler kan het spel starten wanneer hij 5 schepen heeft geplaatst | OK | De speler moet eerst proberen een zesde schip te plaatsen vooraleer de start knop enabled wordt |
| Status van het spel gaat van “nieuw” naar “gestart” | OK | / |
| Na het starten is de start-knop disabled | OK | / |
| Na het starten kan de speler geen schip meer bijplaatsen | OK | / |
| De speler kan een vakje van de tegenstander beschieten | OK | / |
| Beschoten vakjes worden aangeduid | OK | / |
| De speler kan pas aanvallen nadat het spel gestart is | OK | / |
| Als de speler een vakje beschoten heeft waar een schip ligt, wordt dit vakje gekleurd | OK | / |
| Als alle vakjes van een schip beschoten zijn, wordt de kleur van dit vakje aangepast | OK | / |
| Wanneer de speler schiet dan schiet de AI terug | OK | / |
| De score van elke speler wordt getoond en word minder als een van hun schepen word geraakt | OK | / |
| Wanneer alle schepen van een speler zijn gezonken is het spel gedaan | OK | / |
| De winnaar wordt met zijn punten weergegeven | OK | / |
| De speler kan het spel resetten door op de reset knop te drukken die enabled word als het spel eindigt | NOK | Resetten werkt niet naar behoren |

# 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; Spel is observable en de observers zijn de views die geupdate moeten worden. | C:\Users\Faouzi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ObservableUML.PNG |
| Strategy | Ja; de AI voor het plaatsen van schepen en het aanvallen maakt gebruik van het strategy patroon. Een strategy klasse word toegepast aan de hand van de properties.properties file. |  |
| Simple Factory | Ja; Om een schip te maken van een bepaalde soort aan de hand van een String wordt een factory gebruikt die een Enum gebruikt met alle types van schepen die mogelijk zijn. |  |
| Façade | Ja, er wordt een facade gebruik voor alle toegang tot de domain klassen. |  |
| Adapter | Nee | Geen nut om dit patroon te gebruiken |
| Singleton | Nee | Geen nut om dit patroon te gebruiken |
| State | Ja, om bij te houden of een spel gestart is of niet worden de states “gestart” en “nieuw” gebruikt |  |
| Template | Nee | Geen nut om dit patroon te gebruiken |
| MVC | Ja; de controller is de klasse die tussen de view en domain klassen zit |  |

# 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, om bij te houden welk schipsoorten er zijn, hoelang elke schip is en hoeveel een speler van elk schip mag plaatsen |  |
| Enum | Ja, om bij te houden welke richtingen er zijn om de schepen in te plaatsen. |  |
| Properties | Ja, om de strategieën van de ai voor het plaatsen van schepen en het aanvallen toe te kennen | C:\Users\Faouzi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\properties_uml.png |
| Class | Ja, om een object te maken van een strategie in de AI | C:\Users\Faouzi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Class_uml.png |
| JDBC | Nee |  |
| Mockito | Nee |  |
| Streams | Nee |  |
| Lambda-expressies | Nee |  |
| Enz… | Nee |  |

# Werkverdeling

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

|  |  |  |
| --- | --- | --- |
|  | [Naam] | [Naam] |
| Ontwerp |  |  |
| Klassendiagrammen |  |  |
| Implementatie |  |  |
| Verslag |  |  |

# Klassediagramma

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