

**Object Georiënteerd Ontwerp**

**Metro App**

Joossens Benjamin

Lauwers Ernest

Stefanovic Igor

2022-2023 Toegepaste Informatica

29/12/2022

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

# Algemene opmerking

Dit verslag gaat over de opdracht ‘Metro simulatie app’, 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.**

**Je maakt een zip file van al je source code (.java bestanden, geen .class bestanden) en van alle bestanden die van belang zijn voor deze OOO opdracht. Je voegt tevens de laatste versie van je verslag (Word document) toe aan deze zip file.**

**Je laadt deze zip file op via Toledo uiterlijk op donderdag 29/12/2022 - 23.59 uur. Naam van de zip file: volgnr\_familienaam1\_familienaam2\_familienaam3\_familienaam4\_Metro. De familienamen zijn oplopend alfabetisch geordend (vb 17\_Cox\_Dox\_Jansen\_Lenaerts\_ Metro).**

**Het spreekt voor zich dat de code op Toledo overeenstemt met de code in je repository op GitHub.**

# URL GITHUB repository

# https://github.com/Benjamin-Joossens/9\_Joossens\_Lauwers\_Stefanovic\_Metro.git

# 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…

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 en waarom? |
| 1. Overzicht metrokaarten tonen uit txt bestand | ok |  |
| 1. Overzicht metrokaarten tonen met keuze uit tekst of excel | ok |  |
| 1. Metrostation openen | ok |  |
| 1. Nieuwe metrokaart kopen | ok |  |
| 1. Nieuwe ritten kopen | ok |  |
| 1. Metrokaart scannen aan metropoort | ok |  |
| 1. Doorlopen aan metropoort | ok |  |
| 1. Activeren/deactiveren metropoort | ok |  |
| 1. Wijzigen van de settings voor de ritten kortingen | ok |  |
| 1. Afsluiten metrostation | ok |  |

# 

# 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 uit waarom niet.

|  |  |  |
| --- | --- | --- |
|  | Toegepast (ja/nee)  In welke stories(nr)  Waarom toegepast(voordeel) | Bijhorend klassendiagram (gegenereerd vanuit je java code)  GEEN KLASSENDIAGRAM MET ZWARTE ACHTERGRONDKLEUR AFPRINTEN!!!!!!!!!! |
| Observer | Ja, we hebben deze design pattern voor bijna elke story gebruikt. Het Observer patroon maakt het gemakkelijk om nieuwe functionaliteiten toe te voegen of de implementatie van bestaande functionaliteiten te wijzigen zonder dat dit invloed heeft op de rest van het systeem. In ons project zijn al onze controllers observers van de metrofacade. De metrofacade is de subject en notifyt alle ingeschreven observers voor het juiste event. |  |
| Strategy | Ja, we hebben deze design pattern gebruikt bij story 1, 2 en 3. Het Strategy patroon maakt het gemakkelijker om het gedrag van een bepaalde klasse op runtime te kunnen wijzigen. Bij ons project kunnen we dus bijvoorbeeld tijdens dat de applicatie aan het draaien is de loadSaveStrategy aanpassen zonder dat de gebruiker van de applicatie hiervan op de hoogte is. |  |
| Simple Factory | Ja, we hebben deze design pattern gebruikt voor story 1, 2, 3, 5 en 9. Het Simple Factory patroon is nuttig voor het aanmaken van objecten van verschillende types. Het patroon helpt dus om de complexiteit van de code te verminderend door het aanmaken van objecten op een gestructureerde manier te centraliseren in 1 plaats. In ons project gebruiken we 2 keer een factory, onze TicketPriceFactory maakt TicketPrice objecten aan met verschillende korting toegepast. Onze LoadSaveStrategyFactory maakt verschillende LoadSaveStrategy objecten aan. Bv. Excel of Txt. |  |
| Façade | Ja, we hebben deze design pattern gebruikt voor bijna elke story. Het Façade patroon maakt het mogelijk om een complex systeem te hebben dat gemakkelijk te gebruiken is. Door dit patroon hebben we dus de complexiteit van het systeem verborgen achter een enkele, eenvoudige interface. Wat het dus gemakkelijker maakt om met het systeem te werken. In ons project heeft onze MetroFacade allemaal methodes die methodes van de subsystemen aanroepen, waardoor een gebruiker van het systeem dus niet hoeft te weten hoe de subsystemen zijn geïmplementeerd. |  |
| Singleton | Ja, we hebben dit in story 1 gebruikt voor de MetroCardDatabase klasse, dit zorgt ervoor dat er maar 1 instance op deze klasse gemaakt kan worden wat nodig is bij een database van al de metro cards. Dit heeft een voordeel omdat de database klasse over heel de applicatie errors kan geven en er geen aparte managers moeten gemaakt worden. |  |
| State | Ja, we hebben deze design pattern gebruikt in story 6,7 en 8. Het State patroon is handig voor als je het gedrag van een bepaald object bij runtime wilt kunnen aanpassen op basis van hoe het object op verschillende gebeurtenissen reageert. Het helpt dus om complex gedrag te implementeren door het gedrag te verdelen over verschillende staten. In ons project wordt dus het gedrag van een metrogate onderverdeeld in 3 mogelijke statussen. Het gedrag van de poort hangt ervan af of dat de poort actief is of niet en of de poort gescand wordt en of er doorgelopen wordt. |  |
| MVC | Ja, we hebben deze design pattern gebruikt bij elke story. Het MVC-patroon is handig als je de complexiteit van een bepaalde applicatie wil verminderen. Door de gegevens (model), de interface (view) en de logica (controller) van elkaar te scheiden gaat het makkelijker zijn om alles te onderhouden en testen. |  |
| Template method | Ja, dit design pattern hebben we in story 2, 9 gebruikt, dit zorgt ervoor dat we dezelfde soort objecten met verschillende eigenschappen makkelijk kunnen toevoegen met andere waardes. Door dit design pattern te gebruiken kunnen we makkelijk toevoegingen en wijzigingen maken in een object. In ons project kan een loadSaveStrategy klasse de template van bv ExcelLoadSaveTemplate gebruiken om elementen over te nemen. |  |
| Decorator | Ja, we hebben dit design pattern gebruikt in story 9 dit zorgt ervoor dat we een verschillend gedrag van een object dynamisch kunnen toewijzen zonder dat dit iets veranderd aan andere objecten van die decorator klasse.  In ons project hebben we verschillende soorten kortingen die die allemaal toegevoegd kunnen worden door middel van de TicketDiscountDecorator. |  |

# Uitbreidingen App

Leg uit wat er aan je code moet veranderen wanneer we in de toekomst volgende nieuwe functionaliteiten willen toevoegen aan de app (welke nieuwe klassen toevoegen, welke bestaande klassen wijzigen en wat dan wijzigen,…):

|  |  |
| --- | --- |
| Een nieuwe korting voor de rittenprijs toevoegen | De korting moet in TicketPriceDiscountEnum worden toegevoegd. Er moet een klasse worden aangemaakt in de TicketPriceDecorator folder, die TicketPriceDiscountDecorator extends. En de korting moet aan de switch in TicketpriceFactory worden toegevoegd. |
| Een nieuw opslagformaat voor de metrokaarten toevoegen | Om een nieuw opslagformaat toe te voegen aan dit project, moeten we eerst een nieuwe LoadSaveTemplate aanmaken in de utilities folder. Dan in de loadSaveStrategies folder moeten we een nieuwe MetroCardsLoadSaveStrategy aanmaken. Deze klasse gaat dan het gepaste LoadSaveTemplate overerven en de LoadSaveStrategy interface implementeren. In deze klasse moeten we dan alle abstracte klassen Override en we moeten het juiste pad naar het bestand mee geven. Dus als laatste moeten we dan ook nog een bestand van het nieuwe opslagformaat toevoegen in de bestanden folder. |
| Een nieuwe toestand voor een metropoort toevoegen (vb de FREEPASS toestand waarbij de metrogebruikers zonder scannne door de metropoort mogen lopen | Om een nieuwe toestand FREEPASS toe te voegen, moeten we eerst in de metroGateStates folder een nieuwe klasse , genaamd FreePass aanmaken. Deze klasse laten we oververven van MetroGateState. Dan in de MetroGate klasse moeten we FreePass toevoegen als een mogelijke MetroGateState. |

# Speciale topics

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

|  |  |  |
| --- | --- | --- |
|  | Toegepast (ja/nee)  In welke stories(nr)  Waarom toegepast(voordeel) | Bijhorend klassendiagram of extra uitleg |
| Enum | Ja, we hebben het gebruikt voor de verschillende kortingen zodat er makkelijker kortingen toegevoegd kunnen worden en ze direct aan een klasse gelinkt worden. Ook hebben we het toegepast voor de metroEvents en de loadSaveStrategy voor dezelfde reden. |  |
| Properties | Ja, dit hebben we gebruikt voor het opslagen van de actieve kortingen en de loadSaveStrategy zodat het bij de volgende sessie gelezen kan worden met de opgeslagen data. | Een properties bestand is een tekstbestand, dat vaak gebruikt wordt om instellingengegevens op te slaan op een gestructureerde manier. |
| Reflection | Ja, we hebben dit in story 1, 2 en 5 geïmplementeerd. Reflection zorgt voor de flexibiliteit van een klasse. Hierdoor gaat bijvoorbeeld de klasse zichzelf kunnen inspecteren en aanpassen bij runtime. In ons project is dit dus bijvoorbeeld bij het aanpassen van kortingen of bij het aanpassen van de LoadSaveStrategy. |  |
| Andere… |  |  |

# Werkverdeling

Geef aan in percentages hoeveel je bij benadering gespendeerd hebt aan deze opdracht.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Joossens | Lauwers | Stefanovic | Nvt. | Totaal |
| Ontwerp | 50% | 50% | 0% |  | 100% |
| Implementatie | 50% | 50% | 0% |  | 100% |
| Verslag | 50% | 50% | 0% |  | 100% |