

**Object Georiënteerd Ontwerp**

**Kassa App**

[Wout Swennen]

[Jasper Vandenberghen]

[Jeroen Verheyden]

[2019, 2TI SO5-6]

[Datum]

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

# Algemene opmerking

Dit verslag gaat over de opdracht ‘Kassa 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 print dit verslag af (dubbelzijdig) dit in tijdens het herexamen 000**

**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 maandag 23 december 2019 - 23.59 uur. Naam van de zip file: volgnr\_familienaam1\_familienaam2\_familienaam3\_Kassa\_OOO2019. Het volgnr krijg je van de lector.**

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

# URL GITHUB repository

**Copy/paste hier de URL van je Github repository met je zelfevaluatie app project**

# URL

<https://github.com/JasperVandenberghen/28_Swennen_Vandenberghen_Verheyden_KassaApp2019>

# 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 artikelen tonen |  |  |
| 1. Excel bestand inlezen |  |  |
| 1. Registreren kassaverkoop |  |  |
| 1. Tonen kassaverkoop aan klant |  |  |
| 1. Verwijderen artikel uit kassaverkoop |  |  |
| 1. Kassaverkoop on hold |  |  |
| 1. Kortingen toepassen |  |  |
| 1. Kassaverkoop afsluiten |  |  |
| 1. Kassaverkoop betalen |  |  |
| 1. Kassabon printen (op console) |  |  |

# 

# 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) |
| Observer | Story 4, observer toegepast op de artikeldata. |  |
| Strategy | Story 1, strategy toegepast op de Db zodat database makkelijk kan vervangen worden door relationele database. Story 2, toegepast op LoadSave zodat we later ook artikelen kunnen inlezen en wegschrijven met andere bestandsformaten e.d. |  |
| Simple Factory | Factory toegepast voor ArtikelDb en LoadSaveStrategies in story 2. |  |
| Façade |  |  |
| Singleton | Story 2, factories zijn singleton klasses. Om te voorkomen dat een strategy meerde keren aangemaakt moet worden. |  |
| State |  |  |
| MVC |  |  |
| Decorator |  |  |
| Template method | Story 1, template aangemaakt voor TekstLoadSave |  |

# 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, enums aangemaakt met verschillende soorten save strategies, story 2 |  |
| Properties |  |  |
| Reflection | Ja, in de factories maken we gebruik van reflection (en enum) om de juist strategies aan te maken. | Via Class.forName(klasseNaam) maken we met reflection de juiste strategy aan |
| Andere… |  |  |

# Werkverdeling

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | [Familienaam] | [Familienaam] | [Familienaam] | Totaal |
| Ontwerp |  |  |  | 100% |
| Klassendiagrammen |  |  |  | 100% |
| Implementatie |  |  |  | 100% |
| Verslag |  |  |  | 100% |

# Klassen diagramma’s

Voeg het definitieve gegenereerde klassendiagram van je code toe), als afzonderlijk afbeelding-bestand. **Dit klassendiagram moet goed leesbaar zijn.** Spreid het over meerdere pagina’s (bv. 1 blad per package (MVC) en 1 overzicht klassendiagram (zonder attributen en methoden). GEEN KLASSENDIAGRAM MET ZWARTE ACHTERGRONKLEUR AFPRINTEN!!!!!!!!!!