**EINDOPLEVERING**

|  |  |
| --- | --- |
| **Naam** | Timo Brouwer |
| **Studentnummer** | 23055987 |
| **Klas** | 1 |

**OPDRACHT**

Geef hieronder een korte beschrijving van je opdracht (een samenvatting van je productvisie) zodat je docent OPT3 ook begrijpt met welke opdracht jij bezig bent (neem dit onderdeel over uit je laatst ingeleverde opdracht en pas het aan aan de hand van je eigen bevindingen en aan de hand van feedback van medestudenten en je docent aan):

“Freezer" is een app die 65-plussers helpt hun vriezerinhoud te beheren door het scannen van producten en het uitprinten van stickers met QR-codes, namen, en houdbaarheidsdata. Dit vereenvoudigt het bijhouden van voorraden en vermindert voedselverspilling.

**GITHUB**

Maak je GitHub-repo Private en geef je docent voor het werkcollege toegang tot je repo (zie paragraaf 3 in [de uitleg bij deze opdracht op BrightSpace](https://dehaagsehogeschool.sharepoint.com/:b:/r/sites/AgileOOProgrammingSemesterSE-S2_groups-Docenten/Gedeelde%20documenten/Docenten/H-SE-S2OPT3/Lesmateriaal%20(publiek)/OPT3%2014.1%20-%20Realisatie%20zonder%20code%20smells%20-%20Toelichting.pdf?csf=1&web=1&e=MHo6uM) hoe je de repo *clean* houdt en hoe je zorgt dat je een project Maven-*aware* maakt). Stuur niet steeds een nieuwe link/uitnodiging, maar bouw in feature-branches verder aan de applicatie die je eerder hebt opgeleverd:

https://github.com/TimoRoggel/OPT3.git

Bovenstaande repo is m.b.v. .gitignore ontdaan van IntelliJ-files en er is een Maven POM-file toegevoegd waarmee je applicatie in één keer gerund kan worden.

**UML-ontwerp voor OPT2**

Bij deze eindoplevering ben ik uitgegaan van de volgende User Stories (gebruik de ‘+’ rechts van de tabel om een User Story toe te voegen):

|  |  |
| --- | --- |
| **ID** | **Omschrijving** |
| US1 | Als gebruiker wil ik producten kunnen toevoegen aan mijn vriezerinventaris zodat ik het niet meer op papier hoef te schrijven en het niet meer hoef te onthouden. |
| US2 | Als gebruiker wil ik een duidelijk overzicht met producten in de vriezer zodat ik minder voedselverspilling heb. |
| US3 | Als gebruiker wil ik producten kunnen verwijderen nadat ik ze heb gebruikt zodat ik geen dubbele producten koop. |
| US4 | Als gebruiker wil ik een rapportage van mijn voedselvoorraad zodat ik dit kan delen met bijvoorbeeld mijn mantelzorger. |
| US5 | Als mantelzorger wil ik een Admin menu en een normale gebruikersmenu zodat ik de voorraad kan beheren van mijn client zonder dat hij wijzigingen kan toepassen |

**EINDONTWERP**

Voeg hieronder het eindontwerp toe dat je als resultaat hebt opgeleverd bij de vorige opdracht (met eventuele wijzigingen als je dat op basis van feedback van je docent en/of medestudenten wijs leek):

Afbeelding met tekst, schermopname, zwart-wit, ontwerp

Automatisch gegenereerde beschrijving

\* Controleer of het plaatje goed leesbaar is. Als je wilt dat je docent de eerste opdracht voor OPT3 opnieuw beoordeelt, lever je het ingevulde template voor opdracht 1 in via [de inleverbox voor opdracht 1 in BrightSpace](https://brightspace.hhs.nl/d2l/le/lessons/56739/topics/736366).

**EINDOPLEVERING**

Voor de huidige opdracht (de realisatie zonder *code* smells) heb je het eindontwerp uit opdracht 1 (het ontwerp met *patterns* & SOLID) nu omgezet naar werkende code en heb je daaruit de *code smells* verwijderd. Dat laatste is het doel van deze opdracht en daar ligt dus ook het zwaartepunt van de beoordeling van de geschreven code. Hieronder kun je in een checklist nagaan of je alle relevante *Code Smells* uit je code hebt verwijderd en kun je toelichten welke techniek je hebt gebruikt om specifieke *Code Smells* op te ruimen (dat kan dus ook door te verwijzen naar de toepassing van *patterns* en SOLID-principes waarmee je hebt voorkomen dat de *code smells* in je code binnen sluipen).

Zie het volgende document voor voorbeelduitwerking en aanwijzingen:

‘[OPT3 14.1 - Realisatie zonder code smells - Toelichting.pdf](https://dehaagsehogeschool.sharepoint.com/:b:/r/sites/AgileOOProgrammingSemesterSE-S2_groups-Docenten/Gedeelde%20documenten/Docenten/H-SE-S2OPT3/Lesmateriaal%20(publiek)/OPT3%2014.1%20-%20Realisatie%20zonder%20code%20smells%20-%20Toelichting.pdf?csf=1&web=1&e=hLzEIv)’

**CHECKLIST *CODE SMELLS***

Long Method

Large Class

Temporary Field(wordt geen gebruik van gemaakt in mijn code)

Long Parameter List(wordt geen gebruik van gemaakt in mijn code)

Duplicate Code(wordt geen gebruik van gemaakt in mijn code)

Switch-statements(wordt gebruik van gemaakt maar is niet complex waardoor dit geen code smell is)

Primitive Obsession(wordt geen overbodig gebruik van gemaakt waardoor dit geen code smell is)

Divergent Change(wordt geen gebruik van gemaakt in mijn code)

Shotgun Surgery(wordt geen gebruik van gemaakt in mijn code)

**VERANTWOORDING *CODE SMELLS***

Beschrijf hieronder bij de relevante *Code Smells* hoe je ze hebt verwijderd (of waarom ze nog in je code voorkomen):

|  |  |  |
| --- | --- | --- |
| ***Code Smell*** | **Oude situatie** | ***Vervanging*** |
| Large Class | De UserAppMenu-klasse had meerdere verantwoordelijkheden, waaronder het weergeven van het gebruikersmenu, het verwerken van gebruikersinvoer, het afdrukken van productlijsten en het verzenden van lijsten via e-mail. Dit resulteerde in een grote klasse met veel methoden en verantwoordelijkheden. | De verantwoordelijkheden van de UserAppMenu-klasse zijn opgesplitst in kleinere methoden met elk een specifieke taak. Methoden zoals displayUserMenu(), getUserChoice(), printProductList() en sendProductListByEmail() zijn gemaakt om elk een specifiek deel van de functionaliteit van de klasse te behandelen. |
| Door de UserAppMenu-klasse op te splitsen in kleinere methoden met elk een specifieke verantwoordelijkheid, wordt het Single Responsibility Principle toegepast. Elke methode heeft nu een duidelijk afgebakende taak, waardoor de klasse minder complex wordt en gemakkelijker te begrijpen en te onderhouden is. Dit helpt om de Large Class-codesmell te verminderen. | |
|  | |
| Long Methode | De printProductList()-methode was te lang en omvatte meerdere verantwoordelijkheden, waaronder het lezen van productinformatie, het opbouwen van een afdruk, het zoeken naar een printservice, het maken van een afdruktaak en het afhandelen van fouten. | De printProductList()-methode is opgesplitst in meerdere kleinere en meer gespecialiseerde methoden, zoals readProductInfoFromFile(), buildPrintableList(), findPrintService(), createPrintJob() en handlePrintingErrors(). Deze methoden voeren elk een specifieke taak uit binnen het afdrukproces. |
| Door de **printProductList()**-methode op te splitsen in kleinere methoden, wordt het Single Responsibility Principle toegepast. Elke methode heeft nu een duidelijk gedefinieerde verantwoordelijkheid, waardoor de code leesbaarder en onderhoudbaarder wordt. | |