1. INLEIDING

Doorloop de volgende stappen voor het invullen van het template voor een goed UML-Classmodel & realisatie:

1. Bewaar het bestand in een file met je naam en studentnummer.
2. Vul je naam, studentnummer en klas in.
3. Na een toelichting op de onderdelen van het template (met voorbeelden) volgt een Rubric met beoordelingscriteria voor een goed UML-Classmodel & de realisatie daarvan (pagina 5).
4. Als het template jou hindert om een goed ontwerp (met realisatie) te schrijven, vind je op de laatste pagina een korte handleiding hoe je het template ruimer aan kunt passen.

2. OPDRACHT

Voor deze opdracht realiseer je een eerste versie van de software waarin je de productvisie en Product Backlog van H-SE-S2REQS realiseert. Vat dit samen, want je docent OPT2 heeft die nog niet gelezen.

3. GITHUB

Vraag je docent om zijn GitHub-account zodat je hem toegang kunt geven tot je *private repository*.

4. UML-CLASSMODEL

4.1 Inleiding

Over de omvang van de opdracht het volgende: met deze opdracht hoef je alleen maar aan te tonen dat je de leerdoelen van OPT2 (die tot nu behandeld zijn) beheerst. De Rubric op pagina 5 geeft een gedetailleerd beeld van hoe je werk wordt beoordeeld. Hieronder geven we een korte samenvatting van de beoordelingscriteria die leiden tot een goed cijfer:

- Zorg dat alle functionaliteit die je in de User Stories hebt beschreven met het model gerealiseerd kan worden (het model bevat dus de benodigde *Classes* waartussen goede associaties zijn gemodelleerd en waarbinnen alle methodes (met type en parameter(s)) worden opgesomd).
- Het model bestaat uit minimaal 5 *Classes* en er is overerving en *method overriding* toegepast.
- Het is belangrijk dat je de functionaliteit van de *Classes* (wat ga je door welke *Class* laten realiseren?) goed benoemt.
- Het is belangrijk dat je de associaties tussen de *Classes* (met richting en multipliciteit) goed beschrijft, zodat je helder maakt voor welke functionaliteit je die associatie echt niet weg kunt laten.
- Je voegt minimaal 3 zinvolle tests toe (*getters* en *setters* hoef je niet te testen). Een test is zinvol als de complexiteit van de te testen functionaliteit hoog is (ten opzichte van de rest van het programma) en als handmatig testen meer tijd kost dan het schrijven van een geautomatiseerde test.
- De realisatie volgt 1:1 uit het UML-Classmodel, er wordt nergens gebruik gemaakt van een *Godclass* en er is polymorfisme toegepast.

Meestal lukt dat al op basis van 2 of 3 User Stories. Je hoeft dus niet de gehele opdracht te realiseren (dat mag natuurlijk wel). Je ontwerpt en bouwt minimaal dat gedeelte van je opdracht waarmee wij kunnen beoordelen of je de leerdoelen in de Rubric beheerst.

Met het UML-Classmodel neem je een tussenstap die je bespreekt met je docent. Daarin toon je aan dat je een *UML Class Diagram* kunt opstellen en dat je dit model kunt omzetten naar werkende code. Je bespreekt dit model met je docent om te voorkomen dat je met een verkeerd model begint voordat je start met de verwerking van de andere stof van OPT2 en OPT3.

4.2 Casus

Stel bijvoorbeeld dat we uitgaan van de casus met mini's voor DPG Media (die als casus werd gebruikt bij Requirements & Scrum, H-SE-S2REQS). Dat was een project dat qua kosten zomaar in de tonnen €'s liep. Dat is niet het niveau waarop we verwachten dat je voor OPT2 een opdracht uitwerkt. Hoe dan wel?

Met onderstaande User Stories ben ik al in staat om aan te tonen dat ik de stof van OPT2 beheers:

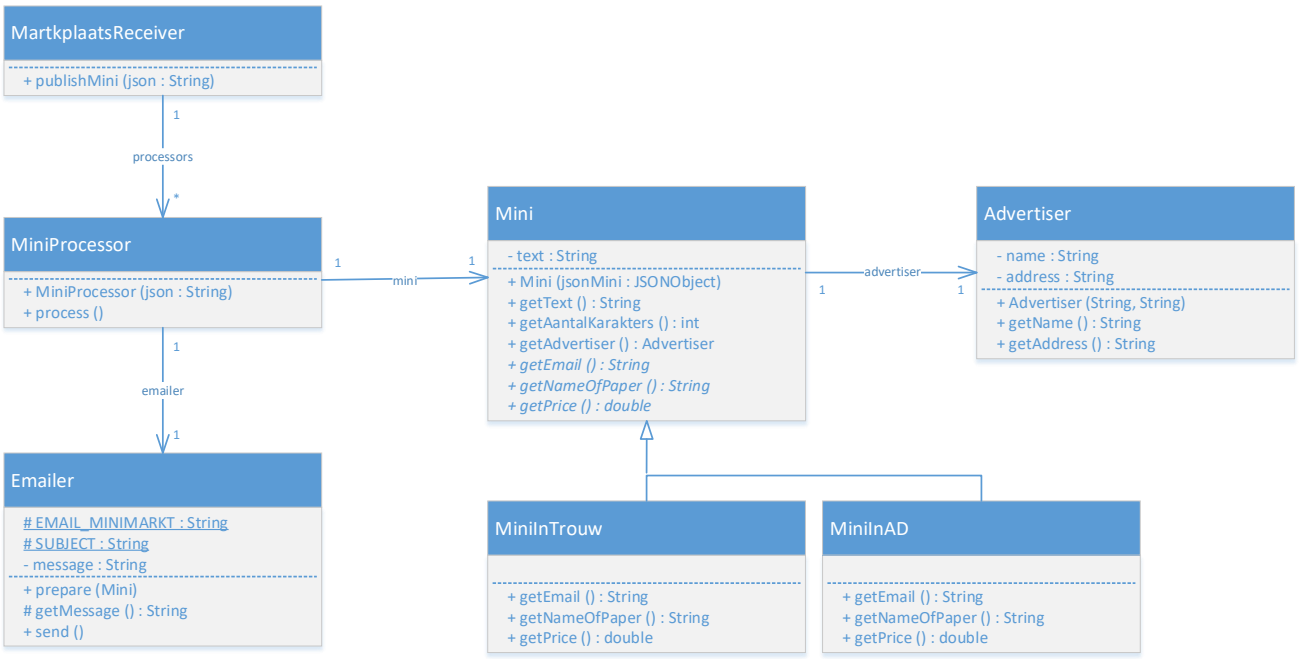
Als medewerker van Trouw wil ik een mail ontvangen als een gebruiker van Marktplaats een mini in Trouw wil plaatsen, zodat ik deze mini kan invoeren in het advertentiesysteem van Trouw en een factuur kan versturen naar de gebruiker van Marktplaats.
Als medewerker van Trouw wil ik graag de prijs van een mini ontvangen als ik de opdracht krijg om een mini te plaatsen in Trouw zodat ik de prijs van een mini in de factuur kan vermelden.

Laten we deze User Stories eens analyseren:

- Stel dat je deze User Story als developer zou moeten realiseren. Dan doe je dat met als doel om de medewerker van Trouw in staat te stellen om de mini in Trouw te plaatsen om daarna een factuur te kunnen versturen. Jij bouwt dus geen software om deze mini te plaatsen of om een factuur te versturen. 'Jouw' software helpt de medewerker van Trouw om dat te kunnen doen.
- Je bouwt wel software om de advertentie vanuit Marktplaats op te kunnen vangen (taak 1).
- Je bouwt wel software om de prijs van zo'n advertentie te kunnen bepalen (taak 2).
- Je bouwt wel software om naam/adres van de adverteerder (voor de factuur) en tekst (voor de advertentie) uit deze aanvraag af te leiden (taak 3).
- En je bouwt tenslotte ook software om deze gegevens te verzamelen in een mail en deze mail te versturen (taak 4).

4.3 Uitwerking

Dan maak ik dus een UML Classmodel waarin alle classes (met onderliggende associaties) voorkomen die ik nodig heb om deze User Stories te kunnen realiseren (dit is het eindresultaat na de stappen hierboven):



* Controleer of het plaatje goed leesbaar is.

Let op de uitwerking: zonder overerving en *method overriding* (en zonder de toepassing van polymorfisme in je realisatie) kun je geen voldoende halen voor deze opdracht.

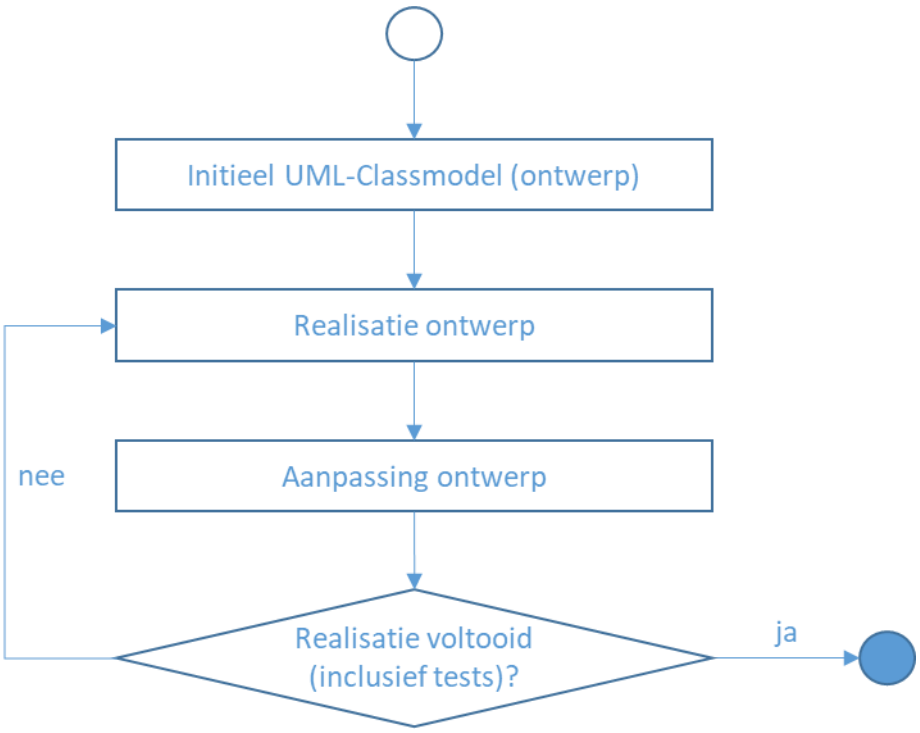
Wat is de functionaliteit van de Classes in het model?

Naam van de Class	**Functionaliteit van genoemde Classes**
MarktplaatsReceiver	Deze Class ontvangt de aanvragen vanuit Marktplaats en geeft opdracht om deze te verwerken.
MiniProcessor	Deze Class verwerkt de gegevens die vanuit Marktplaats worden aangeboden en verzamelt ze in de juiste Classes.
Mini	Deze Class bewaart de tekst van een Mini en kan op basis van die tekst de prijs bepalen.
Advertiser	Deze Class bewaart de gegevens voor een Advertiser.
Emailer	Deze Class verstuurt een email met de juiste gegevens naar de medewerker van Trouw

Naam van de associatie	**Toegevoegde waarde van de genoemde associaties**
processors	Als een advertentie wordt aangeboden vanuit Marktplaats, moet deze worden verwerkt. Er kunnen meerdere verwerkingen tegelijkertijd actief zijn.
mini	Vanuit MiniProcessor wordt een mini aangemaakt, waarvan de gegevens gemaild kunnen worden.
advertiser	Voor elke mini is er één advertiser bekend.
emailer	Het versturen van een mail wordt opgestart vanuit MiniProcessor

6. REALISATIE VAN HET UML-CLASSMODEL

Je doorloopt onderstaande stappen totdat je realisatie 1:1 volgt uit je UML-Classmodel en als de realisatie de functionaliteit in de User Stories correct (dus volgens het beoordelingsmodel) implementeert:



Daarna merge je de code naar de master en copy-paste je de code in je document.

Voor bovenstaand voorbeeld heb ik de realisatie uitgewerkt in https://github.com/kadmosb1/2021MiniNaarTrouw/tree/OPT2.

BEOORDELINGSCRITERIA VOOR UML-CLASSMODEL & -REALISATIE

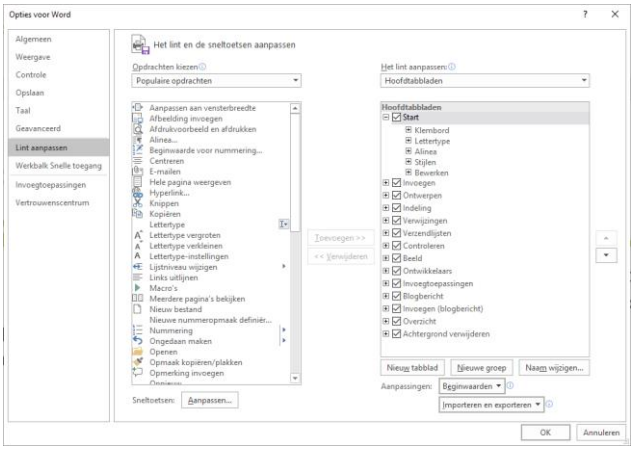
Onderdeel	Niet voldaan (0)	Wel voldaan (10)
Probleemomschrijving (voorwaarde)	Dit onderdeel is voor het starten van de opdracht. Pas als je docent de probleemstelling heeft goedgekeurd, kun je starten met het ontwerp en de realisatie van deze opdracht. De samenvatting van het probleem heeft voldoende omvang en diepgang die nodig is om aan te tonen dat je het vak OPT beheerst (en dus dat je voldoet aan onderstaande voorwaarden). Het beschrijft de oplossing van een probleem dat je in de Productvisie voor REQS hebt omschreven, maar als dat nodig is om het vak OPT te kunnen halen, heb je het (deel-)probleem uit de Productvisie daarvoor aangepast en/of uitgebreid. In de lijst met User Stories zijn alleen die User Stories opgenomen die zijn afgesplitst van Epics en die dus moeten worden gerealiseerd. De User Stories bevatten minimaal 3 zinvolle acceptatiecriteria (zie hieronder voor de voorwaarden voor testen).	
UML Classmodel (30%)	Het model is geen correcte weergave van de structuur van het domein.	Het model is een weergave van de structuur van het domein (incl. correct gemodelleerde en noodzakelijke associaties en voor zover dat wordt omschreven in de gerealiseerde User Stories), bestaat uit minimaal 5 en maximaal 10 zinvolle *Classes,* de methodes staan in de juiste *Classes* en in het model is overerving en *method overriding* zinvol toegepast.
Classes (15%)	De functionaliteit van de *Classes* is niet volledig en/of incorrect uitgewerkt en/of de relatie met de User Stories is niet goed genoeg uitgewerkt.	De functionaliteit van de *Classes* is volledig en correct uitgewerkt en voor elke methode en voor de variabelen in het model is beschreven hoe elk onderdeel bijdraagt aan welke User Story die moet worden gerealiseerd.
Associaties (10%)	Er ontbreken associaties en/of van de associaties is niet volledig en/of incorrect uitgewerkt hoe ze bijdragen aan de User Stories die gerealiseerd moeten worden.	Er ontbreken geen associaties en van elke associatie is volledig en correct beschreven hoe die bijdraagt aan welke User Stor(y/ies) die moet(en) worden gerealiseerd. Er wordt ook beschreven waarom de associatie niet in het model mag ontbreken.
Testen (10%)	Er zijn onvoldoende tests toegevoegd, de tests zijn niet gerelateerd aan acceptatiecriteria en/of de tests testen niet de meest complexe onderdelen van je applicatie.	Er zijn minimaal 3 zinvolle tests toegevoegd die ieder een onderdeel van een andere User Story testen. Deze tests volgen uit de acceptatiecriteria die zijn toegevoegd aan de User Stories en testen de meest complexe onderdelen van je applicatie.
Realisatie (30%)	Het opgeleverde Java-programma werkt niet, komt qua structuur niet 1:1 overeen met het opgeleverde **UML** *Class Diagram*, de functionaliteit is niet volledig uitgewerkt of niet in de juiste methodes en/of classes geïmplementeerd en/of er is geen gebruik gemaakt van polymorfisme.	Het opgeleverde Java-programma werkt, komt qua structuur 1:1 overeen met het opgeleverde **UML** *Class Diagram*, **alle** functionaliteit (die in de lijst met User Stories is beschreven) is in de juiste methodes geïmplementeerd en er is gebruik gemaakt van polymorfisme.

BIJLAGE: HOE KAN IK HET TEMPLATE WIJZIGEN?

We dagen je uit om je producten zo beknopt mogelijk te houden (dat maakt het krachtiger en overzichtelijk; vooral voor degenen die het niet zelf geschreven hebben). Maar soms kan een template je hinderen om je opdracht zo te schrijven dat het past bij jouw ambitie. In dat geval kun je de beperkingen op het template opheffen en alles in het document aanpassen naar je eigen wensen. Maar let op: wij staan er niet voor in dat de huidige opmaak daarbij behouden blijft.

Welke stappen kun je doorlopen om de beperkingen op te heffen?

Activeer het tabblad voor 'Ontwikkelaars' door met je rechter muisknop op het lint te kiezen voor 'Het lint aanpassen…'. In het geopende scherm selecteer je het vinkje voor het hoofdtabblad 'Ontwikkelaars' en kies je voor de knop 'OK':



In het tabblad 'Ontwikkelaars' kies je vervolgens voor de knop 'Bewerking beperken' en kies je daarna voor 'Beveiliging stoppen':