Opdrachten UML Make IT Work

2019-2020

Opmerkingen

1. Voor deze opdrachten heb je een startproject en 1 UML diagram nodig. Die zijn allemaal te vinden op de DLO. Download die files en unzip hem op je laptop. Je krijgt dan de volgende files:
    a. Sequence-Diagram1-Funda.uxf
    b. Opdrachten-OOAD-Startproject.zip
2. Na afloop heb je een aantal documenten: Het aangepaste Intellij project (opdrachten 2+3) en modellen voor opdrachten 1+4+5.
3. Bewaar zowel de **pdf** als de **uxf** files van de modellen in je portfolio (voor de zekerheid)

Context: Startproject beschrijving

Het startproject bevat (een versie van) een Funda programma waar huizen worden aangeboden. Makelaars (die een huis te koop hebben) en (potentiële) kopers kunnen zich registreren op deze Funda-versie. Makelaars zetten huizen erop en kopers kunnen een huis kopen. Het startproject bevat alleen een klein (onvolledig!) deel van het programma om het project begrijpelijk en hanteerbaar te houden. Normaalgesproken zou zo'n programma veel meer functionaliteit bevatten.

Er is een aantal mogelijke use cases gedefinieerd voor dit programma:

- registreren van een makelaar
- registreren van een koper
- registreren van een huis van een makelaar
- het kopen van een huis door een koper
- het opvragen van de totale omzet (verkochte huizen) van een makelaar

Let op, we maken onderscheid tussen de actoren 'Makelaar' en 'Koper' en hun software representaties (de klassen) die in het Funda programma gebruikt worden.

Use case: registreren van een makelaar

Actors: Makelaar
Scenario: Een nieuwe makelaar registreert zich bij Funda door zichzelf aan te melden. Daarbij geeft de makelaar zijn naam en adres. Funda registreert deze makelaar en geeft deze makelaar een unieke id. De makelaar kan deze id gebruiken in het vervolg om zich te identificeren bij Funda.

Use case: registreren van een koper

Actors: Koper
Scenario: Een nieuwe koper registreert zich bij Funda door zichzelf aan te melden. Daarbij geeft de koper zijn naam en adres. Funda registreert deze koper en geeft deze koper een unieke id. De koper kan deze id gebruiken in het vervolg om zich te identificeren bij Funda.

Use case: registreren van een huis van een makelaar

Actors: Makelaar
Scenario: Een makelaar kan een huis registeren bij Funda. De makelaar geeft zijn eigen id en de gegevens van het huis: adres, oppervlakte en prijs. Funda controleert of de makelaar id wel geldig is (een eerder geregistreerde makelaar). Indien de makelaar geregistreerd is, registreert Funda het huis en geeft een unieke id aan dit huis. Bij het huis wordt bijgehouden welke makelaar het op de markt heeft gezet. Kopers en makelaars kunnen in het vervolg het huis id gebruiken om te verwijzen naar dit huis.

Use case: het kopen van een huis door een koper

Actors: Koper
Scenario: Een koper kan een huis kopen bij Funda. De koper geeft zijn eigen id en de id van het huis dat de koper wil kopen. Funda controleert of de koper en het huis wel geregistreerd zijn. Zo ja, dan wordt aan de makelaar van het huis doorgegeven dat het betreffende huis verkocht is. Het huis wordt gemarkeerd als verkocht.

Use case: opvragen van de totale omzet van een makelaar

Actors: Opvrager
Scenario: Een makelaar of een koper kan bij Funda opvragen hoeveel omzet een bepaalde makelaar heeft gemaakt. De opvrager geeft een id van een makelaar en Funda retourneert het totale bedrag van de verkochte huizen die die makelaar heeft verkocht. Er wordt gecontroleerd of de makelaar id wel geregisteerd is.

DLO

Op DLO staat een model: een sequence model (van het registreren van een huis). Download dit model en importeer ze in UMLet.

Er staat ook een startproject op de DLO (`startproject`). In dat project zijn vijf klassen opgenomen. Hiervan is het meeste afgemaakt. Alleen de klasse Funda bevat nog niet geïmplementeerde methodes. Er is ook een unit-test gemaakt. Bestudeer de code en zorg dat je ze begrijpt.

De omschrijving van de opdracht en de modellen zijn voldoende gedetailleerd om een werkend project te krijgen. Een voorbeeld unit-test is meegeleverd.

Opdrachten

Opdracht 1

Maak een UML klassendiagram voor het Funda project. Gebruik daarbij de code uit het startproject als basis. *Hint*: UMLet kan een deel van het klassendiagram genereren uit java-code.

- Je hoeft geen klassen te tekenen voor klassen uit de standaard bibliotheken van Java (List, HashMap, etc).
- Je mag alle getters/setters/constructor/toString methodes uit het UML klassendiagram weglaten, die spreken voor zich.
- Vergeet de juiste associaties, cardinaliteiten, attributen, *visibilities* en notaties voor static, abstract, etc. niet!

Opdracht 2

In `Sequence-Diagram1-Funda.pdf` is een sequence diagram gegeven van de use-case "het registeren van een huis". Implementeer de code voor de bijbehorende methode `registreerHuis(…)` in `Funda.java`.

Opdracht 3

De use case "het kopen van een huis" is nog niet gemodelleerd of geïmplementeerd. Implementeer de methode `koopHuis(…)` in `Funda.java`, aan de hand van de use case beschrijving. Hint: `Funda.java` bevat alle Maps die je nodig hebt om deze use-case te implementeren.

Opdracht 4

De use case "het opvragen van de totale omzet van een makelaar" is al geïmplementeerd. Modelleer het gedrag van de methode `totaleOmzet(…)` in `Funda.java` in een UML Sequence Diagram. Licht, indien nodig, je antwoord kort en bondig toe.

Opdracht 5 (Geen startproject nodig)

Beschouw een simpel computer spel: een Fighting Game. In zo'n spel vechten twee on-screen karakters tegen elkaar. Een speler kan een karakter besturen door middel van het drukken van knoppen op een 'game console'. In dit spelletje kan het karakter verschillende houdingen aannemen: springend, staand of knielend. Verder kan het karakter verschillende vechtbewegingen maken, afhankelijk in welke houding het karakter staat. Er zijn 4 knoppen waarmee de speler zijn karakter kan besturen: up, down, fight, special. Welke actie wordt gedaan bij welke knop, wordt in onderstaande beschrijving gegeven:

- Bij het indrukken van de 'up' knop, verandert het karakter van houding: van knielend naar staand en van staand naar springend. Een karakter kan continu blijven springen in dit spel.
- Bij het indrukken van de 'down' knop verandert het karakter van houding: van springend naar staand, van staand naar knielend.
- Bij het indrukken van de 'fight' knop, doet het karakter een aanval. In de knielende houding, doet hij een vuistslag, in de staande houding doet hij een schop, in de springende houding doet hij een dubbele trapbeweging.
- Bij het indrukken van de 'special' knop, doet het karakter iets bijzonders. In de knielende houding, maakt hij een koprol en eindigt in de staande houding; in de staande houding, maakt hij een blokbeweging, in de springende houding maakt hij een duikbeweging en eindigt daarna in de knielende houding.

Maak een UML State diagram voor deze Fighting Game. Gebruik het blikpunt van de softwareontwikkelaar die het spel moet implementeren. Bedenk goed wat de "toestanden" zijn en wat de "events" zijn die je van de ene toestand naar de andere toestand doet overgaan. Laat in je State diagram ook de acties zien. Laat in je diagram *alle* mogelijke events zien die een karakter kan doen in elke state, ook de ogenschijnlijk "nutteloze". Kies logische "start" en "eind" toestanden (indien van toepassing) en geef deze aan.