GSO3 oefentoets voorjaar 2014

A.u.b. drie gegevens invullen voordat je aan de opgaven begint!

|  |  |
| --- | --- |
| naam |  |
| klas |  |
| studentnr |  |

Deze GSO3-toets bestaat uit twee opgaven

De antwoorden op de vragen moeten in de voorgeschreven bestanden worden opgenomen.

|  |  |
| --- | --- |
| Opgave | Antwoord in |
| 1 | Bijgeleverd Visual Paradigm project |
| 2a | Bijgeleverd Visual Paradigm project |
| 2b | Bijgeleverd NetBeans project |
| 2c | Dit opgaveformulier (Word-document) |

Opgave 1

Ontwerp een klassendiagram van het domein met alle vereiste attributen en associaties in VP. Elke klasse, interface en attribuut heeft een naam. Elke associatie die geen compositie is heeft een associatienaam of ten minste 1 rolnaam. De navigatie van een associatie wordt niet gevraagd. Elke associatie heeft aan elke uiteinde een multipliciteit. Andere constraints hoef je niet te vermelden. Afleidbare informatie hoef je in het klassendiagram niet op te nemen.

Gegeven is het volgende domein

In een huisartsenpraktijk werkt een aantal huisartsen. Een huisarts kan een gewone huisarts zijn, een gespecialiseerde huisarts, of een huisarts in opleiding (stagiair). Elke huisarts heeft een aantal patiënten, die bij hem of haar op de raadpleging komen. Per raadpleging maakt de huisarts een aantekening en mogelijk schrijft hij een recept uit met een aantal medicijnen. Een recept kan eenmalig zijn of een herhalingsrecept waarmee dezelfde medicijnen een op te geven aantal keren kan worden verkregen. Verder is van ieder medicijn bekend wat de minimum leeftijd van de patiënt moet zijn om het medicijn te mogen gebruiken en of het medicijn generisch is, danwel of er een generische variant beschikbaar is (en zo ja, welke). De recepten worden doorgestuurd naar de apotheek van de patiënt. In elke apotheek is wel een variant van ieder medicijn verkrijgbaar. In sommige gevallen kan een huisarts een patiënt doorsturen naar een van de ziekenhuizen waarmee de praktijk een contract heeft voor verder onderzoek. Het onderzoeksresultaat wordt door het ziekenhuis teruggekoppeld aan de huisarts, die het verwerkt in een aantekening bij het dossier van de patiënt.

Het klassendiagram hoeft alleen de mogelijkheid te bieden om de informatie van één huisartsenpraktijk te herbergen. De klasse Praktijk is gegeven (klassendiagram Opgave\_1). Zorg ervoor dat deze klasse Praktijk in je ontwerp een rol van betekenis vervult.

Opgave 2

ValueHub is een klassestructuur waarmee verschillende applicaties benoemde waardes kunnen delen. Hieronder zijn de belangrijkste functionele requirements opgesomd:

|  |
| --- |
| Action   1. Een applicatie kan een client registreren bij de hub. 2. Een applicatie kan bij een client een benoemde waarde instellen. 3. Een applicatie kan bij een client een benoemde waarde opvragen. 4. Een applicatie kan zich bij een client registreren om berichten te ontvangen als een waarde verandert. |
| RelevantInformation   1. Een benoemde waarde heeft een naam en een waarde |
| Rule   1. Elke benoemde waarde kan bij elke client worden opgevraagd. 2. Alle benoemde waarden zijn lokaal bij elke client beschikbaar. 3. Zodra een client zich aanmeldt bij de hub, worden alle reeds bekende waarden uitgewisseld, waarbij de waarden van de hub leidend zijn. |

Voor deze requirements is een software-ontwerp gemaakt. Zie VP-project Opgave2. Het ValueHub-singleton-object is verantwoordelijk voor zich laten registreren van ValueClient objecten en het het doorgeven van de NamedValue objecten aan alle aangemelde clients. Een ValueClient-object draagt zorg voor het versturen van nieuwe benoemde waarden naar de hub en het ophalen van benoemde waarden voor één applicatie. Zodra een applicatie wordt aangemeld bij de hub, worden alle al bekende waarden uitgewisseld, waarbij de waarden op de hub leidend zijn.

Opgave 2a

De eerste action requirement gaat over het aanmelden van een client. We houden hierbij rekening met de derde rule requirement. Er wordt een sequencediagram gevraagd waarmee het klassendiagram wordt getoetst en moet worden verbeterd. Het aanmelden van een client kan als één scenario worden beschouwd. Dit is het moment waarop het te ontwerpen sequence diagram begint:

De applicatie geeft de ValueClient opdracht zich aan te melden bij een ValueHub.

De ValueClient meldt zich aan bij de ValueHub.

De hub doorloopt zijn lijst met NamedValues en geeft die door aan de nieuwe client.

De ValueClient geeft al zijn waarden door aan de ValueHub.

Tijdens het melden van waarden door de hub bij de client, worden eventueel geregistreerde IValueChangedListeners over de nieuwe waarden geïnformeerd. Deze acties behoren niet tot het scenario.

In het VP-project (zie RegisterClient) is een sequentiediagram als startpunt aangeboden. Werk in dit sequentiediagram het scenario verder uit. Als blijkt dat het gegeven klassendiagram (Opgave\_2) niet afdoende is, word je dringend uitgenodigd om de correctie(s) in dat klassendiagram aan te brengen.

Opgave 2b

Er wordt een unit test gevraagd. Gegeven zijn de javadoc-specificaties van de methoden setValue in de klasse ValueClient en setValue in de klasse ValueHub.

public class ValueClient {

…

/\*\*

\* Zet de waarde behorende bij name op value.

\* Als er bij deze waarde nog geen naam was

\* geregistreerd, wordt een nieuwe waarde opgeslagen.

\* Als er wel al een waarde bekend was, wordt de waarde

\* gewijzigd. In alle gevallen wordt de waarde

\* doorgegeven aan de hub waarbij de client is

\* geregistreerd, zodat de (nieuwe) waarde ook aan andere

\* aangesloten clients wordt doorgegeven.

\* @param name : de naam waarvan de waarde wordt

\* gewijzigd.

\* @param value : de nieuwe waarde voor deze naam

\*/

void setValue(String name, String value);

…

}

public class ValueHub {

…

/\*\*

\* Zet de waarde behorende bij nv.name op nv.value.

\* Als er bij deze waarde nog geen naam was

\* geregistreerd, wordt een nieuwe waarde opgeslagen.

\* Als er wel al een waarde bekend was, wordt de waarde

\* gewijzigd. In alle gevallen wordt de waarde

\* doorgegeven aan alle clients die bij de hub zijn

\* geregistreerd.

\* @param nv : het naam-waarde paar dat wordt gewijzigd.

\*/

void setValue(NamedValue nv);

…

}

De gegeven unittest ValueClientTest hoef je alleen met betrekking tot onderstaande twee aspecten, op basis van bovenstaande javadoc-specificaties, aan te vullen.

1. Worden de waarden inderdaad opgeslagen als ze nog niet bestonden en worden bestaande waarden correct aangepast?
2. Schrijf zelf een test die nagaat of de ValueHub inderdaad de waarden verspreid onder de verschillende IValueListener objecten die zijn aangemeld.

Opgave 2c

Het klassendiagram Opgave\_3 dient als basis voor een gedistribueerd systeem. Het idee is dat de ValueHub op een centrale machine gaat draaien en de ValueClient objecten op verschillende andere machines. De ValueHub en de ValueClient objecten communiceren conform het RMI-protocol.

Extending een klasse en implementing een interface kun je beide onder overerving scharen. Geef voor onderstaande klassen door middel van een kruisje (x) aan of ze van Remote, Serializable of UnicastRemoteObject moeten overerven. Een regel hoeft niet per se een kruisje te bevatten. Evenzo is het toegestaan om, indien gewenst, op een regel twee of drie kruisjes te plaatsen.

Uitwerking 2c: Overerving

|  |  |  |  |
| --- | --- | --- | --- |
|  | Remote | Serializable | Unicast-  RemoteObject |
| IValueListener |  |  |  |
| ValueHub |  |  |  |
| ValueClient |  |  |  |
| NamedValue |  |  |  |
| IValueChangedListener |  |  |  |