Braindump Softwareentwicklung in Großprojekten WS 19/20

17. Juni 2020 (wegen Corona erst im SS20)

Aufgabe 1

Überprüfen Sie die Richtigkeit der folgenden Aussagen und erklären Sie bei den falschen warum sie falsch sind.

a) Um gute Wartbarkeit zur erreichen, sollte Kohäsion und Kopplung hoch sein.

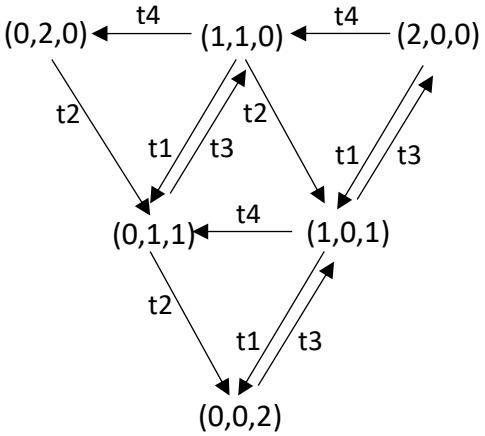
b) Agile Softwareentwicklung ist gut für Sicherheitssysteme.

c) Ein Error führ nicht unbedingt zu einem Versagen.

d) Whitebox-Testing findet gleiche bzw. mehr Fehler als Blackbox-Testing.

e) Ein Kommunikationsdiagramm in ein Sequenzdiagramm umzuwandeln, ist möglich.

Aufgabe 2



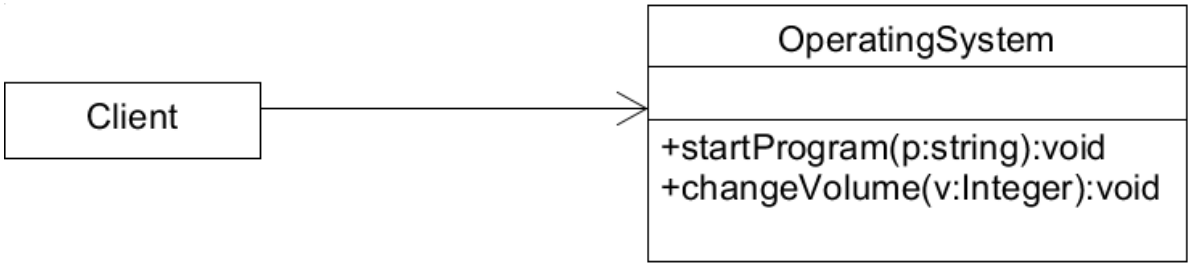
Wandeln Sie diesen Erreichbarkeitsgraphen in ein Petrinetz um.

Aufgabe 3

Den Hotkeys einer Gaming Tastatur sollen durch den Benutzer verschiedene parametrisierte Aktionen zugewiesen werden (z.B. Systemlautstärke um einen vorgegebenen Wert ändern oder das Öffnen eines Programms). Die Zuordnung soll zur Laufzeit bestimmt werden.

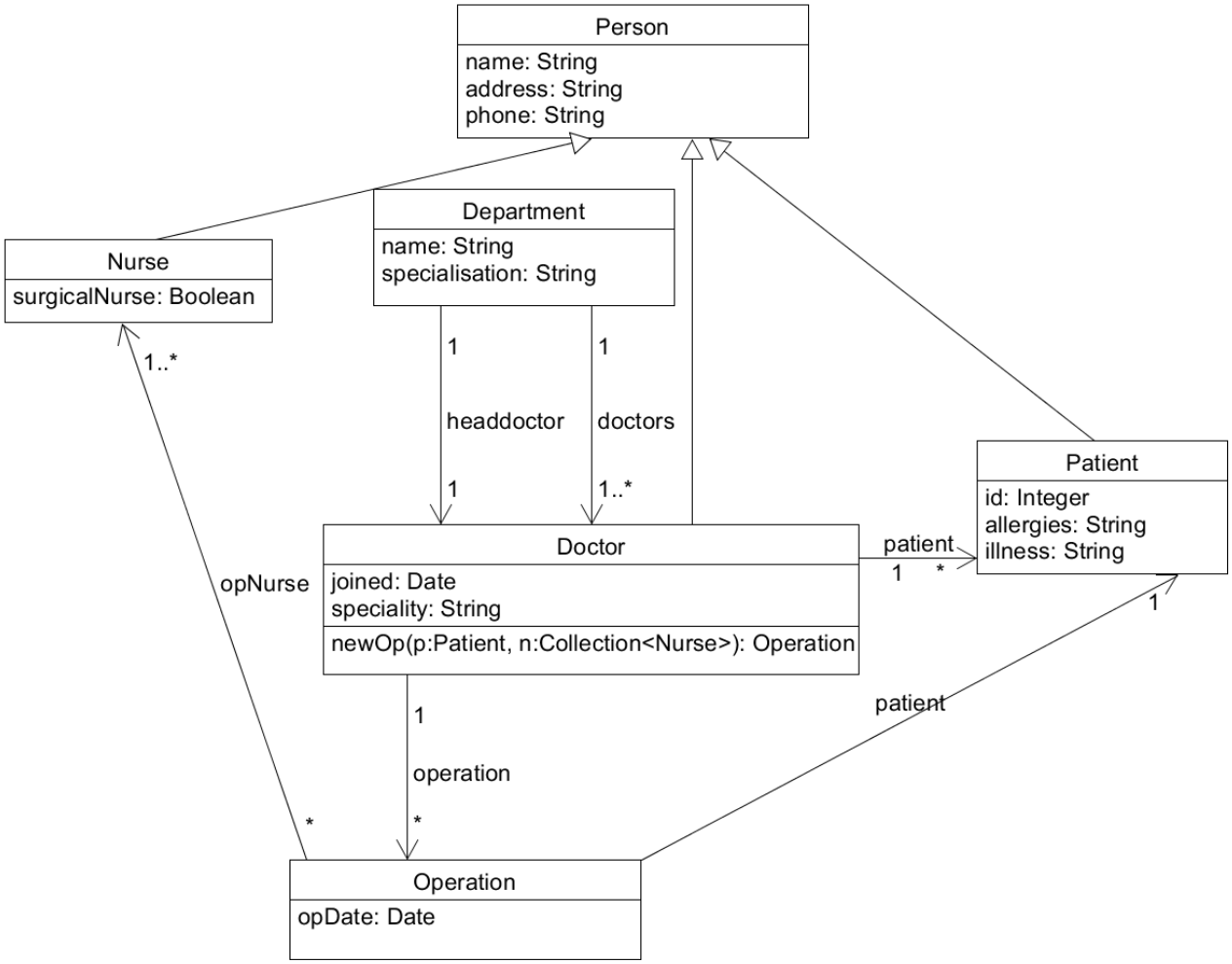
a) Welches Entwurfsmuster eignet sich am besten dafür?

b) Vervollständigen Sie das folgende UML-Diagramm:



c) Zeichnen Sie ein Sequenzdiagramm für den Fall, dass der Benutzer einen Hotkey drückt, für den die Aktion *C:\Programm\MsWord.exe öffnen* hinterlegt ist.

Aufgabe 4



Formulieren Sie folgende Anforderungen in OCL:

a) Der Chefarzt ist auch Arzt im Department

b) Jeder Arzt betreut max. 5 Patienten

c) Patienten haben eindeutige Ids

d) Der Aufruf Doctor.newOp(p, n) setzt voraus, dass jede Krankenschwester in n eine Op-Krankenschwester ist. Nach dem Aufruf ist die neue Operation dem Arzt zugeordnet.

Aufgabe 5

```
public abstract class Discount{
    public abstract double getValue();
}

public class WinterDiscount extends Discount{
    @Override
    public double getValue(){
        return 0.10;
    }

    public String description(){
        return "Your disount:"+100*getValue()+"(December to February only)";
    }
}
public class SummerDiscount extends Discount{
    @Override
    public double getValue(){
        return 0.20;
    }

    public String description(){
        return "Your disount:"+100*getValue()+"(June to August only)";
    }
}
public class AdultWinterDiscount extends WinterDiscount{
    @Override
    public double getValue(){
        return 0.05*super.getValue();
    }
}
public class ChildWinterDiscount extends WinterDiscount{
    @Override
    public double getValue(){
        return 0.02*super.getValue();
    }
}
public class AdultSummerDiscount extends SummerDiscount{
    @Override
    public double getValue(){
        return 0.05*super.getValue();
    }
}
public class ChildSummerDiscount extends SummerDiscount{
    @Override
    public double getValue(){
        return 0.02*super.getValue();
    }
}
```

a) Nennen Sie drei aus der Vorlesung bekannte Arten des Refactorings, die auf diese Klasse sinnvollangewendet werden können.
b) Zeichnen Sie ein UML-Klassendiagramm, welches das Softwaresystem nach dem Refactoring beschreibt.