Aufgabe 1: Theoriefragen

a)

Entscheiden Sie, ob diese Aussagen korrekt oder inkorrekt sind. Begründen Sie ihre Antwort.

1. Wird bei einem strukturellen (White-Box) Test eine vollständige Pfadüberdeckung erreicht, so schließt dies auch eine vollständige Verzweigungs- und Anweisungsüberdeckung ein.
2. Ein Modul hat kommunikative Bindung, falls es eine Reihe von Aktionen auf gemeinsamen Daten ausführt. Die Reihenfolge der Aktionen ist dabei von Bedeutung.
3. Unter Refactoring versteht man die Änderung des Funktionsumfangs.

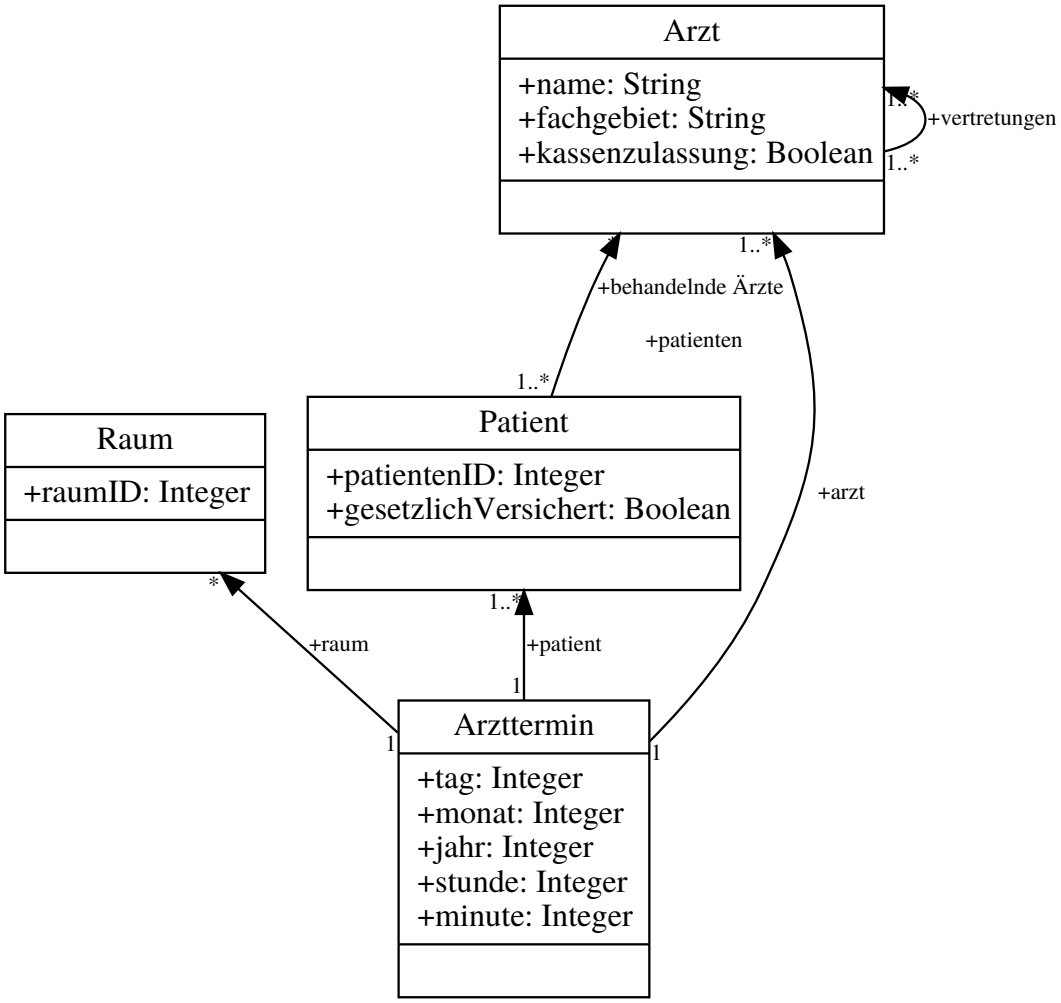
b)

Definieren Sie die folgenden Begriffe:

1. Validierung
2. Datenstrukturkopplung
3. Vorbeugende Wartung

Aufgabe 2: OCL

Gegeben sei das folgende Klassendiagramm in UML:



Formulieren Sie die folgenden Anforderungen in OCL:

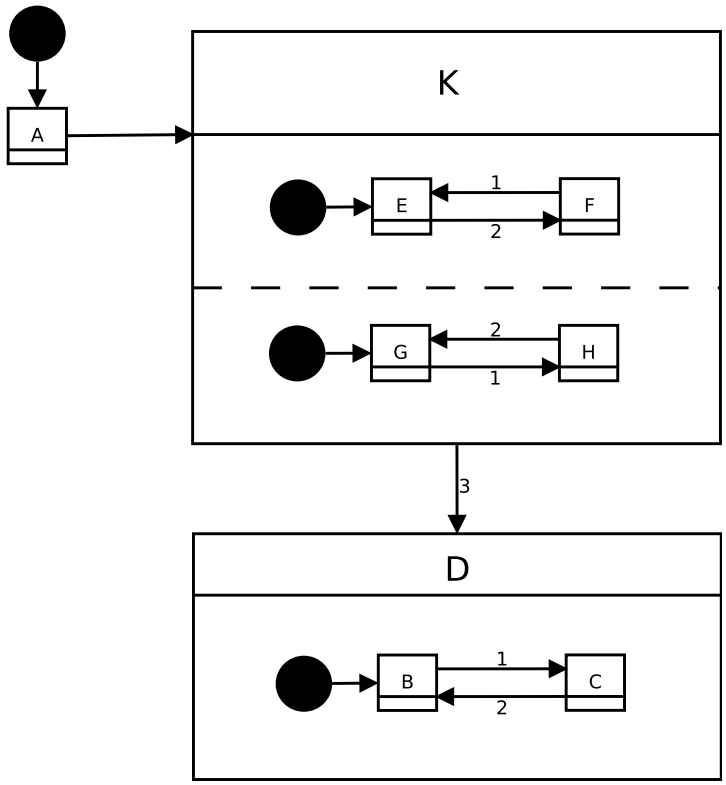
1. Im Raum 107 dürfen keine Arzttermine stattfinden.
2. Jeder Arzt hat mindestens eine Vertretung, die im selben Fachgebiet tätig ist.
3. Jeder Arzt darf die Vertretung von höchstens drei Kollegen übernehmen.
4. Ein Patient darf von verschiedenen Ärzten behandelt werden, jedoch müssen diese alle ein anderes Fachgebiet aufweisen.
5. Ein gesetzlich versicherter Patient darf nur von einem Arzt mit Kassenzulassung behandelt werden.
6. Zwischen zwei Arztterminen beim selben Arzt liegen mindestens zehn Minuten.

Aufgabe 3: Entwurfsmuster

Eine Klassenbibliothek stellt eine Klasse `File` mit seiner Methode `save()` zum Speichern der jeweiligen Datei zur Verfügung. Die Bibliothek soll nun um eine Benachrichtigungsfunktion erweitert werden, die interessierte Benutzer über das Speichern informiert.

1. Welches aus der Vorlesung bekannte Entwurfsmuster eignet sich zur Realisierung?
2. Welche Klasse und welcher Gültigkeitsbereich ist diesem Entwurfsmuster zuzuordnen?
3. Zeichen sie ein Klassendiagramm, um diesen konkreten Fall darzustellen.

Aufgabe 4: State-Chart



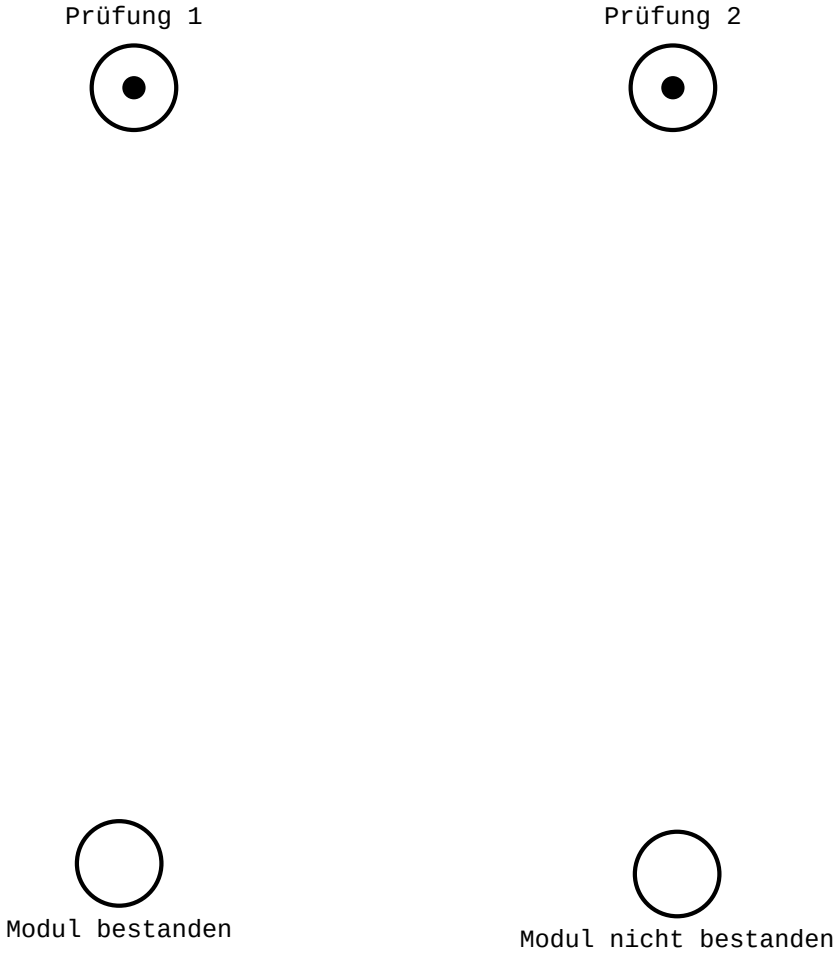
1. Wandeln Sie den gegebenen State-Chart in einen entsprechenden endlichen Automaten um.
2. Welche der Zustände werden maximal einmal erreicht?

Aufgabe 5: Petrinetze

Ein Prüfungsmodul besteht aus Teilprüfungen *Prüfung 1* und *Prüfung 2*. Das Modul gilt genau dann als bestanden, wenn beide Teilprüfungen bestanden wurden. Eine nicht bestandene Teilprüfung wird höchstens zweimal wiederholt. Sobald eine Teilprüfung beim dritten Versuch nicht bestanden wurde, gilt das gesamte Modul als nicht bestanden.

Modellieren Sie das beschriebene Verhalten, indem Sie das nachfolgende Petrinetzgerüst um Plätze, Tokens, Transitionen, Kanten (inkl. Inhibitor-Kanten) so erweitern, dass folgende Eigenschaften erfüllt sind:

- Bei bestandener Prüfung wird eine Deadlockmarkierung mit mindestens einem Token in den Platz *Modul bestanden* und kein Token in den Platz *Modul nicht bestanden* gelegt.
- Bei nicht bestandener Prüfung genau anders herum.



Aufgabe 6: Testen

Gegeben sei folgender Java-Code:

```
int function(int a, int b, int c) {
    int result = 0;

    if (a == c) {
        b--;
    }

    if (a + b + c <= 6) {
        if (a + b == 4) {
            result = b - c;
        } else {
            result = b + c;
        }
    }
    return result;
}
```

Geben Sie die minimale Anzahl an Eingaben (i, j, k) mit $i, j, k \in \{0, 1, 2, 3, 4, 5\}$ an, so dass...

1. Vollständige Anweisungsüberdeckung erreicht ist.
2. Vollständige Verzweigungsüberdeckung erreicht ist.
3. Vollständige Pfadüberdeckung erreicht ist.