Umbruch von Rafael
Version 1 11.03.20

WS 18/19 09.04.2019

Koziolek

Softwaretechnik 2

Dauer: 90 min. Lösung: keine Bestanden mit: ? P.
Bemerkung: Nachklausur

Aufgabe 1: Verschiedenes (17 Punkte)

a) Nennen Sie zwei Vorteile eines iterativen Vorgehens gegenüber des klassischen Wasserfallmodells. (2 Punkte)

b) Nennen Sie drei *View Points* in Palladio und beschreiben Sie diese kurz. (4.5 Punkte)

c) Für den Softwareentwurf werden in den GRAS-Mustern Aussagen über Kopplung (*Coupling*) und Kohäsion (*Cohesion*) gemacht. Sollten Kopplung und Kohäsion jeweils hoch oder niedrig sein? Begründen Sie Ihre Entscheidung. (3 Punkte)

d) Beschreiben Sie kurz den Unterschied zwischen Coding und Refactoring. Gehen Sie dabei auch kurz darauf ein, wie sich die Ergebnisse von Testfällen in beiden Fällen verhalten sollten. (3 Punkte)

e) Nennen Sie drei *Test Double Patterns*, mit denen man Code isoliert testen kann. Erläutern Sie außerdem kurz die jeweilige Funktionsweise. (4.5 Punkte)

Aufgabe 2: Anwendungsfälle (15 Punkte)

Folgende Beschreibung charakterisiert die zugrundeliegende Funktionalität eines Zimmerbuchungssystems für Hotels:

Wenn ein Gast oder Hotelmitarbeiter ein Zimmer buchen möchte, dann wird zunächst die Zimmerverfügbarkeit der gewünschten Zimmerkategorie überprüft und bei einem verfügbaren Zimmer ein Zimmer der gewählten Kategorie reserviert. Beim Check-in eines Gastes ruft der Hotelmitarbeiter die entsprechende Reservierung im System auf und initialisiert dort den Check-in. Sollte keine Reservierung vorhanden sein, dann muss der Hotelmitarbeiter die Zimmerverfügbarkeit überprüfen und kann dann den Check-in für eine Zimmerkategorie mit freien Zimmern initialisieren. Beim Check-in weist das System dem Gast ein freies Zimmer der gewünschten Kategorie zu und gibt die Information, welches Zimmer zugewiesen wurde, an den Hotelmitarbeiter zurück. Sollte bei einem Gast mit einer Reservierung dennoch kein Zimmer in der gewünschten Kategorie mehr verfügbar sein, dann wird der Gast auf die nächsthöhere Zimmerkategorie mit freien Zimmern heraufgestuft und in dieser Kategorie ein Zimmer zugewiesen. Dabei wird das Zimmer im System mit einer entsprechenden Markierung versehen. Die Zimmerzuordnung wird abgeschlossen, indem der Hotelmitarbeiter den Gast informiert und das zugewiesene Zimmer bestätigt. Daraufhin kann der Hotelmitarbeiter eine Schlüsselkarte für das Zimmer dem Gast übergeben. Beim Check-out wird anhand der Übernachtungen und des Preises für das Zimmer der Rechnungsbetrag ermittelt und die Bezahlung daraufhin abgewickelt. Am Ende der Bezahlung wird dem Gast eine Rechnung übergeben und das Zimmer im System für die Endreinigung markiert. Nachdem die Endreinigung abgeschlossen wurde, kann das Zimmer im System wieder als frei verfügbar markiert werden.

a) In der Vorlesung haben Sie drei Shaping-Heuristiken zur Identifikation guter Anwendungsfälle (*Use-cases*) kennengelernt. Benennen und beschreiben Sie kurz diese drei Heuristiken. Geben Sie auch ein Beispiel aus dem Beschreibungstext an, das nach diesen Heuristiken kein guter Anwendungsfall ist. (3.5 Punkte)

b) Verwenden Sie das textuelle Anwendungsfallbeschreibungsschema *Fully-Dressed* zur vollständigen Spezifikation des Anwendungsfalls „Check-In“. Nutzen Sie hierfür die unten gegebene Schablone. Geben Sie auch zwei Erweiterungen an. (8.5 Punkte)

Name d. Anwendungsfalls *(Use Case Name)*	
Geltungsbereich *(Scope)*	
Primäraktor(en) *(Primary Actor(s))*	
Stakeholder(s)	
Vorbedingungen *(Preconditions)*	
Nachbedingung *(Postconditions)*	
Primäres Erfolgsszenario *(Main Success Scenario)*	
Erweiterung (Alternativer Ablauf) *(Extension; Alternative Flow)*	

c) Nennen Sie die vier *Use Case Goal Levels* nach Cockburn. Ordnen Sie den Anwendungsfall aus Teilaufgabe b) einer der Stufen zu. Begründen Sie dabei kurz Ihre Entscheidung. (3 Punkte)

Aufgabe 3: Modellierung (15 Punkte)

a) Eine komplette Spezifikation eines Metamodels besteht aus vier Bestandteilen. Ein Bestandteil ist die abstrakte Syntax. Erklären Sie diesen Bestandteil näher. Nennen Sie außerdem die anderen drei Bestandteile. (4 Punkte)

b) Erstellen Sie das Klassendiagramm eines *Metamodells* für die Domäne des im Folgenden beschriebenen Flugticketsystems. Berücksichtigen Sie Modellklassen, Beziehungen, Rollen, sowie Multiplizitäten. (8 Punkte)

Im Flugticketsystem können Flugtickets sowie Kunden verwaltet werden. Flugtickets gelten jeweils für eine bestimmte Flugverbindung. Eine Flugverbindung besteht immer aus einem Startflughafen und einem Zielflughafen, wobei ein Flughafen immer einen Namen und ein Kürzel besitzt. Ein Flugticket gilt für eine bestimmte Abflugzeit und besitzt einen Preis. Ein Flugticket ist auch immer entweder ein Economy-Class-Ticket, ein Business-Class-Ticket oder ein First-Class-Ticket. First-Class-Tickets besitzen stets einen Zugangscode, mit dem man in die First-Class-Lobby kommt. Ein Ticket ist immer einem Kunden als Fluggast zugeordnet, wobei ein Kunde einen Namen und ein Geburtsdatum hat. Um treue Kunden zu belohnen, können diese auch an einem Vielfliegerprogramm teilnehmen. Beim Vielfliegerprogramm gibt es auch das spezielle Vielflieger-Business-Programm, bei dem der Kunde besondere Belohnungen erhält, dafür jedoch nur Business-Class- und First-Class-Tickets angerechnet werden. In beiden Programmen werden die Tickets gespeichert, wobei im Vielflieger-Business-Programm entsprechend nur Business-Class- und First-Class-Tickets gespeichert werden können.

Hinweis: Datentypen (wie etwa int, long, Date,...) müssen **nicht** angegeben werden.

c) Warum ist Ihre Lösung von Teilaufgabe b) ein Modell? Benutzen Sie die Definition eines Modells von Stachowiak und setzen Sie diese in Bezug zu den Attributen Ihrer Lösung. (3 Punkte)

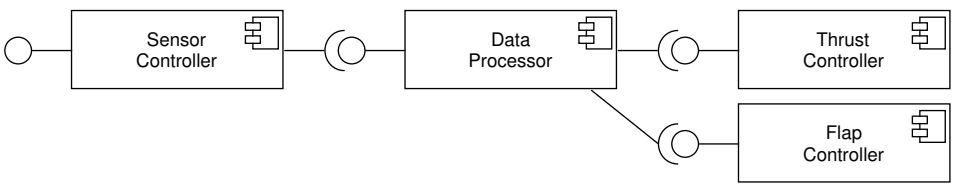
Aufgabe 4: Echtzeitsysteme (12 Punkte)

Betrachten Sie folgendes Szenario: Sie entwerfen ein Assistenzsystemen für die Steuerungstechnik eines Flugzeugs. Das System unterstützt den Piloten bei der Navigation, dem Start und der Landung. Es greift aktiv in die Regelung der Steuerung, des Schubs (*Thrust*), sowie der Klappen (*Flaps*) des Luftfahrzeugs ein.

a) Um welches der beiden Arten von Echtzeitsystemen handelt es sich bei dem Assistenzsystem? Begründen Sie Ihre Antwort. (2 Punkte)

b) Welche Risiken birgt eine nicht redundant ausgelegte Komponente? Schildern Sie einen Risiko-Fall, der in dem gegebenen Szenario auftreten kann. Unter welchen Bedingungen kann das Problem dabei auftreten? (2 Punkte)

c) Erweitern Sie das folgende Komponentendiagramm so, dass sie überprüfen können, ob Sensordaten in einem bestimmten (gültigen) Intervall liegen. Welches Muster wenden Sie an? (5 Punkte)



d) Warum reicht es bei einem solchen Steuerungssystem nicht immer zu prüfen, ob sich Werte in einem gültigen Rahmen bewegen? Welches Muster sollten Sie bei besonders kritischen Systemen, wie dem Steuerungssystem, verwenden? Kosten spielen nun keine Rolle. (3 Punkte)

Aufgabe 5: Software-Architektur (14 Punkte)

a) Wie lautet die Regel, mit der sich zwischen einem OO-Design-Pattern (wie dem Singleton-Pattern) und einem Architekturmuster unterscheiden lässt? (2 Punkte)

b) Die Performance von Software-Komponenten hängt gemäß Palladio von vier Einflussfaktoren ab. Neben dem eigentlichen Code, haben Hardware und aufgerufene externe Dienste wesentlich Einfluss auf die Performance. Nennen Sie den vierten Einflussfaktor, erklären Sie seine Bedeutung und geben Sie ein Beispiel an, an dem die Rolle des Einflussfaktors deutlich wird. (4 Punkte)

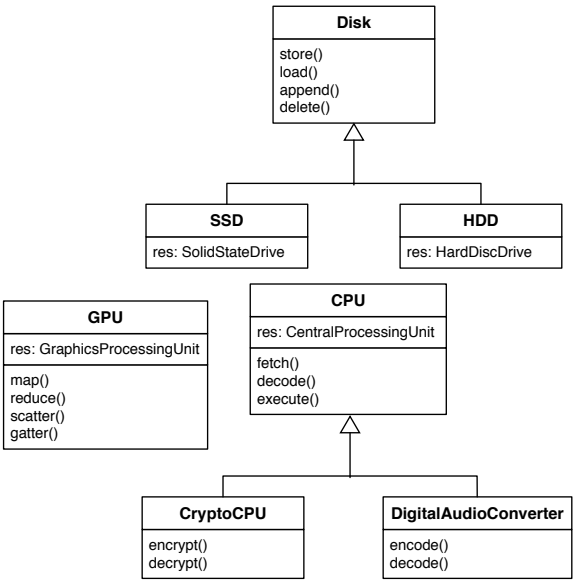


Abbildung 1: Vereinfachtes Klassendiagramm für *ResourceAccess*

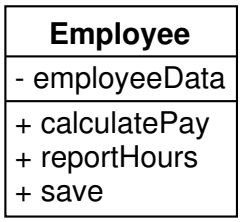
c) Sie entwerfen eine Software-Architektur, die ein Ressourcenverwaltungs- und Ressourcenzuweisungssystem für Rechenressourcen realisieren wird. Ihre Architektur enthält eine Komponente für den Zugriff auf verschiedene Ressourcentypen wie z.B. CPU, GPU, HDD, die dem Benutzer Zugriff auf die verfügbaren Ressourcen gewährt. Möchte der Benutzer eine Ressource nutzen, so instanziiert er die entsprechende Klasse (im Klassendiagramm dargestellt) innerhalb der Komponente und arbeitet mit Hilfe der Methoden, welche von der Klasse angeboten werden, auf der Ressource. Das vereinfachte Klassendiagramm der Komponente *ResourceAccess* könnte wie in der Abbildung dargestellt aussehen (das genaue Verständnis der Klassen ist nicht nötig und dient nur zu Illustrationszwecken). Das Problem dabei ist, dass jeder Aufrufer Ressourcen nach belieben (durch Instanziierung der Klassen) allokieren kann. Benennen Sie ein Pattern, welches das Problem beheben kann. Erklären Sie, wieso das Pattern hierbei hilft. (3 Punkte)

d) Was sind *Smart UIs*, warum sollte man sie vermeiden und wie kann man das Problem beheben? Was erreicht man letztendlich mit der Lösung? (3 Punkte)

e) Was sind Referenzarchitekturen? Beschreiben Sie diese kurz und nennen Sie einen Vorteil der Verwendung von Referenzarchitekturen. (2 Punkte)

Aufgabe 6: SOLID-Prinzipien (9 Punkte)

Gegeben sei die Klasse für einen Mitarbeiter (`Employee`) aus der Abrechnungsanwendung in Unternehmen. In dieser gibt es drei Methoden: `calculatePay`, `reportHours` und `save`.



Diese Methoden sind für drei sehr unterschiedliche Akteure zuständig:

- Die Methode `calculatePay` wird von der Buchhaltungsabteilung spezifiziert und dient der Lohnberechnung des Mitarbeiters.
- Die Methode `reportHours` wird von der Personalabteilung spezifiziert und verwendet und dient der Stundenabrechnung des Mitarbeiters.
- Die Methode `save` wird vom Datenbankadministrator spezifiziert und dient der Datenspeicherung des Mitarbeiters.

a) Gegen welches der fünf SOLID-Prinzipien verstößt diese Klasse? Geben Sie den vollständigen Namen des Prinzips an und begründen Sie, warum diese Klasse gegen dieses Prinzip verstößt. (3 Punkte)

b) Beschreiben Sie, wie man die Modellierung anpassen könnte, um den Verstoß zu beheben. Die Lösung soll gegen kein SOLID-Prinzip verstoßen und dennoch den gegebenen fachlichen Anforderungen gerecht werden. Zeichnen Sie dazu auch ein UML-Klassendiagramm ihrer Lösung. (6 Punkte)

Aufgabe 7: Clean Code (8 Punkte)

```
class User {
      int group;
      boolean isSenior;
}
```

```
import java.util.*;

class Application {

      int getId(final User x) {
            // Test, ob der Benutzer eine Admin ist
            if (((x.group <= 13) && x.isSenior) || (x.group >= 37)) {
                  // Wenn ja, erhält er eine durch die Formel ermittelte positive ID
                  return Math.abs((x.group * 42) << 42);
            } else {
                  // Wenn nicht, erhält er eine durch die Formel ermittelte negative ID
                  return -Math.abs((x.group * 42) << 42);
            }
      }

      void printUsers(final ArrayList<User> users) {
            for (final User user : users) {
                  System.out.println(user);
            }
      }
}
```

a) Betrachten Sie in dem obigen Java-Quelltext die Methode `printUsers()`. Benennen und erklären Sie in der Signatur dieser Methode einen Verstoß gegen die Praktiken des guten Codings (Clean-Code-Prinzipien und Code Conventions aus dem Clean-Code-Vorlesungsfoliensatz) enthalten. Geben Sie eine angepasste Signatur für diese Methode an, um den Verstoß zu beheben, die dennoch den gegebenen fachlichen Anforderungen gerecht wird. Eventuell auftretende Probleme im Programmablauf (Exceptions o.ä.) können hierbei ignoriert werden. (2 Punkte)

b) Betrachten Sie in dem obigen Java-Quelltext die Methode `getId()`. Benennen und erklären Sie in dieser Methode vier Stellen im Quelltext, welche unterschiedliche Typen von Verstößen gegen die Praktiken des guten Codings (Clean-Code-Prinzipien und Code Conventions aus dem Clean-Code-Vorlesungsfoliensatz) enthalten. Eventuell auftretende Probleme im Programmablauf (Exceptions o.ä.) können hierbei ignoriert werden. (6 Punkte)