Gedächtnisprotokoll Klausur Progammierparadigmen vom 22.07.2020

Allgemeine Hinweise:

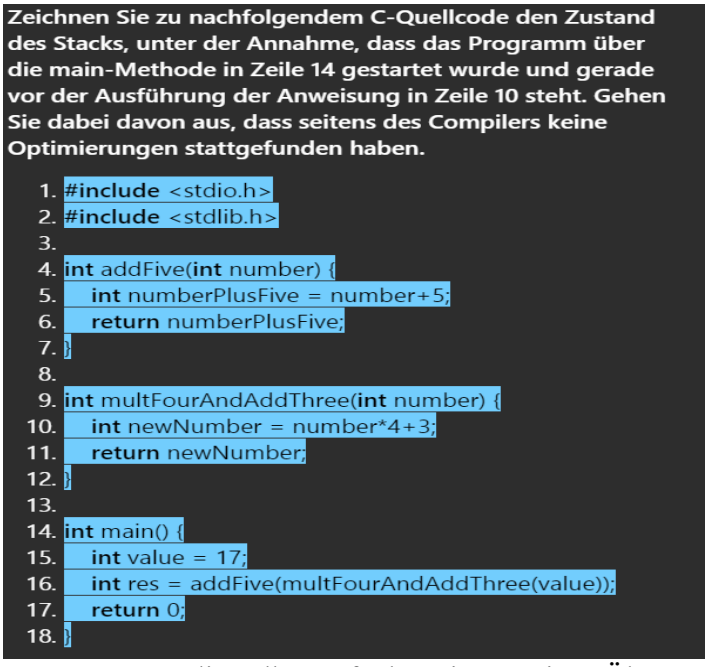
- Kein Anspruch auf Vollständigkeit!!! Ich bin mir sicher, daß ich noch ein paar Aufgaben vergessen habe hier aufzuschreiben.
- Reihenfolge ist nicht dieselbe wie in der Klausur!!!!!
- Das nochmalige Lösen der Übungsaufgaben hat gereicht, um für die Klausur ausreichend vorbereitet zu sein(durchfallen sollte man dann jedenfalls nicht)
- Trotzdem nochmal die Folien durchgehen, um Themen zu lernen, die nicht in der Übung drankamen(z.B. Aufgabe 3d wurde in keiner Übungsaufgabe behandelt)
- Es gab keine Aufgaben, wo irgendetwas programmiertechnisch umgesetzt werden sollte
- Auswendiglernen muss man eig. nichts. Konzepte der Paradigmen sollte man verinnerlicht haben
- Paralleles Paradigma war nicht Thema, da es nicht behandelt wurde
- Es waren 21 Seiten und sehr viele Aufgaben. Man hat im Prinzip keine Zeit sich die einzelnen Regeln und Sachverhalte nochmal in Ruhe herzuleiten. ->erst die Aufgaben mit den meisten Punkten machen und schnell zur nächsten Aufgabe, wenn man länger hängenbleibt

Aufgabe 1 Prozedurales Paradigma

a)
call-by-referene vs call-by-value + Vorteile/Nachteile

b) Prinzipien des Prozeduralen Paradigmas und Kernidee(Wiederverwendbarkeit/Kapselung des Codes usw)

c)



(Achtung!: Das war die selbe Aufgabe wie aus einer Übungsaufgabe(Bild) mit dem wichtigen Unterschied, daß in der Klausuraufgabe „**nach** der Ausführung der Anweisung in Zeile 10“ steht anstatt wie hier „**vor**“. Also genau lesen!)

d)Was macht Reference Counting Garbage Collection und wann würde man es anwenden?

e)
Thema Heartbleed Bug in der SSL-Verschlüsselung. Es wurde kurz das Prinzip von Heartbeat-Nachrichten zwischen Client und Server erklärt, welche die Verbindung am Leben erhalten sollen. Es war ein C-Codeschnipsel gegeben wie dies im Prinzip umgesetzt ist.

Es sollte anhand des Codes erklärt werden, was die Heartbleed-Attacke hier eigentlich ist und wie man sie beheben könnte.

(Heise hat die Thematik schön erklärt: https://www.heise.de/security/artikel/So-funktioniert-der-Heartbleed-Exploit-2168010.html)

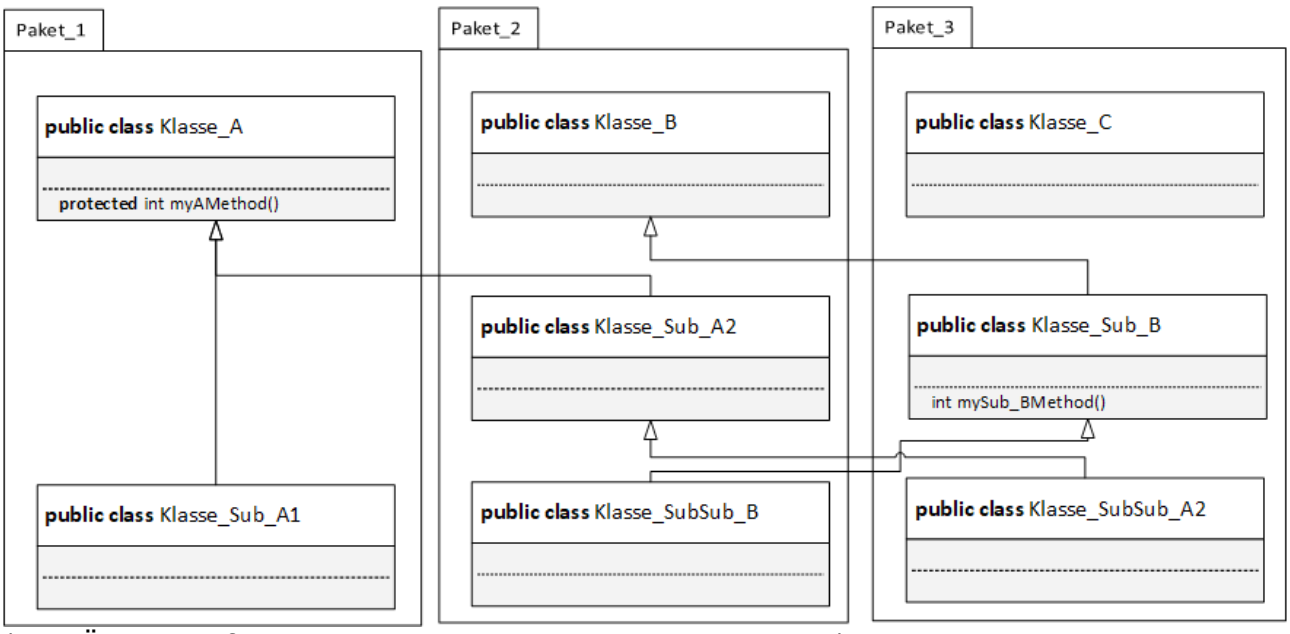
Aufgabe 2 Objektorientiertes Paradigma(OOP)

a) Java verkörpert in reinster Form das OOP(sinngemäß mit meinen Worten). Ist die Aussage richtig? Anhand den Sprachelementen von Java erklären.(Konzepte der OOP anhand von Java zeigen).

b) In der Abbildung ist eine Vererbungshierarchie in der Form eines UML-Diagramms zu sehen.

Geben Sie für jede der acht dort angegebenen Klassen an, ob diese jeweils auf die Methode *myAMethod* bzw. *mySub_BMethod* zugreifen kann.

Hinweis: Für die Kapselung wurde in dem UML-Diagramm jeweils die zugehörige Syntax aus der Sprache Java verwendet.



(War Übungsaufgabe und kam 1zu1 so in der Klausur dran)

c) Aufgabe zur Linearisierung/Vererbungshierarchie. In jedem Schritt erklären, welche Regeln verwendet wurden.

Aufgabe 3 Funktionales Paradigma

a)Prinzipien der Funktionalen Programmierung erklären(Kein Kontext/Seiteneffekte, usw)

b)Funktionen höherer Ordnung erklären. Welche Rolle spielen dabei Unterversorgung und Currying. + 2 Anwendungsbeispiele

c)Java Code gegeben, welches nicht dem funktionalen Paradigma folgt: Erklären an welchen Stellen es Verletzungen gibt und Korrigierungsmöglichkeiten.

d)Es waren 4 oder 5 Scala Codeausschnitte gegeben, die Pattern Matching implementieren. Für jeden Ausschnitt entscheiden, ob Pattern Matching disjunkt und/oder erschöpfend ist.

e)Es waren Lambda Terme gegeben, welche vereinfacht werden sollten. Dabei jeden Vereinfachungsschritt angeben!

f)
Es war ein längerer Prosatext gegeben. Was ich noch in Erinnerung habe:

- Firma verleiht Fahrräder
- Fahrräder können an Knotenpunkten innerhalb der Stadt ausgeliehen werden
- Nach Verleih wird alle x Sekunden Geschwindigkeit, Ort usw gespeichert
- All diese Werte dienen für persönliche/globale Bestlisten und auch für die

Rechnung danach
- Server erstellt intern statischen Graph über alle Fahrräder und deren Aufenthaltspunkten
- QR Codes gab es auch noch

Es sollte argumentiert werden, wo hier das Funktionale Paradigma sinnvoll einsetzbar wäre und wo vllt. weniger.

g) Was ist Lazy Evaluation und wie kann man es in Scala anwenden?

h) Wie kann man induktiv Datentypen/Funktionen definieren? Was ist dafür notwendig? (Stichwort Scala Streams)

Aufgabe 4 Logisches Paradigma

a) Was ist Unifikation? Regeln nennen, unter welchen Umständen zwei logische Ausdrücke unifizierbar sind.

b) Prolog Programm gegeben.
Die Aussagen und Regeln des Programmes sollten in Hornformeln/definite Klauseln umgewandelt werden.
SLD-Resolution durchführen für eine gegebene Anfrage und beim ersten Ergebnis aufhören. Es sollte Tiefensuche verwendet und die Regeln der Reihenfolge nach abgearbeitet werden.
(Es war hier einmal Backtracking notwendig. Unifikatoren mit angeben!)

War ähnlich zu folgender Aufgabe aus der Übung, d.h. Backtracking und Anwenden von Zahlvergleichen usw

Gegeben sei folgendes Prolog-Programm. Schreiben Sie das Programm als Hornausdruck gemäß der Darstellung in der Vorlesung. Führen Sie die SLD-Resolution mit der folgenden Anfrage durch, wobei Regeln mittels Tiefensuche gemäß der Reihenfolge im Programm gefunden werden. Gehen Sie davon aus, dass bei mehreren Lösungen die erste gültig ist (D.h. es kann die Resolventenbildung abgebrochen werden).

Hinweis: Innerhalb der Hornklauseln stellt ',' (das Komma) das logische Oder dar, während in Prolog das ',' das logische Und darstellt.

?− nF(adenauer,X). *(Lösung: X=erhard)*

```
1.%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2.% wK - warKanzlerIn
3.% re - regiert
4.% nOG - nachOderGleich
5.% nF - nachfolger
6.%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7.
8.wK(adenauer, 1949,1963).
9.wK(erhard, 1963,1966).
10.wK(kiesinger, 1966,1969).
```

```
11.wK(brandt, 1969,1974).
12.wK(schmidt, 1974,1982).
13.wK(kohl, 1982,1998).
14.wK(schroeder, 1998,2005).
15.wK(merkel, 2005,2020).
16.
17.re(K,J):- wK(K,A,E),nOG(J,A),nOG(E,J).
18.nOG(J1,J2) :- J1 >= J2.
19.nF(K,NF) :- wK(K,_,E),wK(NF,A,_),nOG(A,E).
```

Kontakt zum Autor: p.seidel-qlb@web.de