Gedächtnisprotokoll Moderne Programmiertechniken & Methoden

Prüfer: Prof. Dr. Steimann
Beisitzerin. Dr. Keller
Datum: 21.10.15
Note: 1,3

Vorbereitung:
Zur Vorbereitung habe ich die einzelnen Kapitel zusammengefasst und dabei versucht alles sehr genau nachzuvollziehen. Zum lernen habe ich dann die Zusammenfassungen auswendig gelernt und mir Karteikarten mit groben Stichpunkten erstellt. Anhand derer habe ich dann zur Wiederholung des Stoffs einen Vortrag gehalten.

Prüfung:
Zunächst wurde ich gefragt, mit welchem Thema ich beginnen möchte. Ich habe Interfacebasierte Programmierung gewählt und nach einer etwas wirren Einleitung meinerseits habe ich vor allem die Arten des Gebrauchs von Interfaces erläutert. Dadurch kamen wir dann auf das Thema Blackbox- und Whitebox-Frameworks. In diesem Zuge ging es dann weiter zu JUnit und wie dieses Framework einzuordnen ist (durch das Template-Pattern: Whitebox). Dann fragte Prof. Steimann welche Patterns sonst noch in JUnit eingesetzt werden. Ich nannte das Composite Pattern, kam aber nicht auf das Listener Pattern. Als nächstes wurde nach dem Role Object Pattern gefragt. Ich erklärte es und nannte die Objektschizophrenie als eines der Probleme hierbei. Dann fragte Prof. Steimann, wo Objektschizophrenie noch vorzufinden ist. Da kam ich dann natürlich auch nicht drauf, aber es lief auf das Refactoring „Vererbung durch Delegation ersetzen“, hinaus. Das erläuterte ich dann und Prof. Steinmann lenkte dann um zum Thema Metaprogrammierung, auf die Frage, was für ein Metaprogramm ein Refactoringwerkzeug ist.

Fazit:
Es lohnt sich wirklich, diesen Kurs zu bearbeiten, ich habe viel dabei gelernt.
Sowohl Prof. Steinmann als auch Dr. Keller sind sehr nett und sehr gute Prüfer. Ich kann sie beide nur weiterempfehlen!
Im nach hinein denke ich, dass meine Vorbereitung zwar gut war, ich allerdings kurze, 5-minütige Vorträge hätte üben sollen, da ich mich teilweise entweder in Details oder komplett den Faden verloren habe.

Gedächtnisprotokoll zur Video-Prüfung 01853 (Januar 2015) bei Prof. Dr. Steimann:

Prüfungsvorbereitung:
Zur Vorbereitung auf die Prüfung habe ich zu jedem der Kurskapitel einen kleinen Vortrag vorbereitet. Mein Schwerpunkt lag dabei auf den Themen Metaprogrammierung, Design by Contract und Unit-Testing. Außerdem habe ich mir aus den Protokollen, den Übungsblättern und den alten Klausuren Fragelisten zu jedem Kapitel erstellt, die ich in regelmäßigen Abständen wiederholt habe.

Prüfungsumfeld:
Diese Prüfung war meine zweite Video-Prüfung bei Prof. Steimann. Zu Beginn der Prüfung wurde ich in den Medienraum des Studienzentrums gebeten, vor meinem Sitzplatz befanden sich zwei Monitore. Auf dem einen Monitor waren Prof. Steimann und die Beisitzerin Dr. Keller zu sehen, auf dem anderen Monitor ich selbst. Bei beiden meiner bislang absolvierten Video-Prüfungen empfand ich es zunächst als sehr unnatürlich und unangenehm über Mikro und Kamera zu kommunizieren (und mich dabei auch noch selbst im Großformat zu sehen), im Laufe der Prüfung habe ich dank der ausgezeichneten Video- und Ton-Qualität und der Größe der Monitore aber immer beinahe vergessen, dass es sich um eine Video-Prüfung und nicht um eine Prüfung vor Ort handelt. Insofern kann ich es Studierenden, die weit weg von Hagen wohnen, nur empfehlen, das Angebot der Video-Prüfung wahrzunehmen, das Prof. Steimann kulanter Weise auch für Studierende anbietet, die nicht im Ausland leben.

Prüfungsablauf:
Prof. Steimann fragte mich, mit welchem Thema ich beginnen möchte. Ich wählte die Metaprogrammierung und hielt ca. 7 Minuten ohne Unterbrechung meinen dazu vorbereiteten Vortrag. Im Anschluss stellte Prof. Steimann einige Fragen:

- Sie haben gerade Refactoring-Tools als Metaprogramme erwähnt. Was ist den das Besondere an Refactoring-Tools? (Zunächst wusste ich nicht genau, worauf Prof. Steimann hinauswollte, schließlich kamen wir aber zu dem Punkt, dass ein Refactoring-Tool das Verhalten des Programms nicht verändern darf.)
- Erklären Sie das Refactoring „RIWD“: Was macht dieses Refactoring? Was ist der Unterschied zwischen Delegation und Forwarding? Was sind die Vorbedingungen zu diesem Refactoring?
- Was ist eine Vorbedingung, die jedem Refactoring gemeinsam ist? (Hier stand ich zunächst wieder auf dem Schlauch. Scheinbar wollte Prof. Steimann hier hören, dass das zu refaktorisierende Programm kompilierbar (oder kompiliert?) sein muss.)
- Sagt Ihnen der Begriff „Objektschizophrenie“ etwas? (Antwort: Äh, nein. Hier wusste ich leider überhaupt nicht, was Objektschizophrenie im Zusammenhang mit Refactorings verloren hat. Prof. Steimann ritt nicht allzu lange auf meiner Unwissenheit herum und leitete über zum Thema Design by Contract.

Daraufhin hielt ich einen rund 5-minütigen Vortrag zum Thema DbC (abermals ohne Unterbrechung) und im Anschluss bat Prof. Steimann ohne weitere Rückfragen die Betreuerin im Studienzentrum, den Ton abzuschalten. Nach ca. einer Minute meldete sich Prof. Steimann zurück und teilte mir mit, dass ich die Prüfung sehr gut abgeschlossen habe.

Fazit:
Ich kann die mündlichen Prüfungen bei Prof. Steimann uneingeschränkt empfehlen. Nach meiner bisherigen Erfahrung bietet es sich an, Vorträge vorzubereiten und möglichst viel frei vorzutragen (dabei sollte man sich auch nicht irritieren lassen, wenn Prof. Steimann während des Vortrags die Augen geschlossen hält). Oft hatte ich bei den konkreten Fragen, die Prof. Steimann stellte, Probleme zu verstehen, worauf die Frage genau abzielt – allerdings gab es auch keinen Notenabzug, wenn ich eine Frage nur halb oder auch gar nicht beantworten konnte.
Als konkreten Tipp möchte ich noch mitgeben, nicht das Thema Extreme Programming als Einstiegsvortrag zu wählen, sondern sich intensiv in eines der komplexeren Themen des Kurses einzuarbeiten und dieses ausführlich vorzutragen.

01853 Moderne Programmiertechniken und -methoden

Prüfer: Prof. Steimann

Beisitzer: Frau Dr. Keller

Datum: 07.10.2013

Vorbereitung:

Ich habe den Kurs im Semester mit sämtlichen Übungsaufgaben bearbeitet. Vor der Prüfung habe ich mir zu jeder Kurseinheit (KE) kleinere Vorträge erstellt, welche den Kurs zusammengefasst haben. Jeder Vortrag umfasste 10 bis 18 Folien (Vorträge zu KE 4 und 5 waren relativ kurz, weil ich mir nur die wesentlichen Essenzen der einzelnen Entwurfsmuster und Refaktorisierungen eingeprägt habe). Da Professor Steimann am liebsten den Kandidaten frei vortragen lassen möchte, halte ich diese Vorbereitungstechnik für sinnvoll.

Prüfungsablauf:

Die Prüfung begann mit der Frage, mit welchem Thema ich beginnen möchte. Ich habe mich für die *Metaprogrammierung* entscheiden, weil diese KE sich m E. gut strukturiert vortragen lässt. Ich habe die KE 6 vollständig in 15 Minuten vorgestellt, ohne dass Prof. Steimann eine Zwischenfrage gestellt hat. Aus seiner Miene kann man auch nicht ablesen, ob er mit den Ausführungen zufrieden war. Ich habe versucht Querbezüge zu anderen Themen des Kurses zu schaffen, beispielsweise auf das Unit Testing und die dynamisch konfigurierbaren Systeme (Stichwort *Assembler* KE 1). Eigene Erfahrungen, wie z. B. die Verwendung der Metaprogrammierung in C++ (Präprozessor, Templates) habe ich ebenfalls ganz kurz genannt.

Da noch Zeit war, sollte ich etwas zum *Design by Contract* erzählen. Aufgrund der fortgeschrittenen Zeit ging es nur bis zu den Zusicherungen. Danach kamen wir noch kurz auf das Thema Refaktorisierung. Er wollte explizit etwas zu den Vorbedingungen wissen, ich habe ihm die Vorbedingungen *von replace Inheritance with delegtion* aufgezählt.

Ende der Prüfung:

Nachdem die Zeit rum war hat er mich für etwa 30 Sekunden rausgeschickt. Er sagte, dass die Prüfung ihm sehr gut gefallen und er mir eine 1,0 gibt. Danach haben wir noch 5 Minuten über meinen Beruf und mein Interesse an dem Studiengang Praktische Informatik gesprochen, was ich sehr sympathisch fand.

Fazit:

Der Kurs war wirklich sehr interessant, sollte aber m. E. nur von Leuten mit einer gewissen Programmiererfahrung gemacht werden. Ansonsten erkennt man die Notwendigkeiten für den Einsatz der vorgestellten Techniken nicht unbedingt. Die Prüfung wurde mehr als fair bewertet, da ich glaube, dass ich nicht alles ganz sauber vorgetragen habe. Ich denke, dass Prof. Steimann Wert darauf legt, dass die Leute sich mit dem Thema intensiv auseinandergesetzt haben.

Prof. Steimann kann man auf jeden Fall als Dozenten und Prüfer uneingeschränkt weiterempfehlen!

Viel Erfolg!

Prüfungsprotokoll

26. September, 2013
Kurs 01853 - Moderne Programmierkonzepte

Die Prüfungen fanden zeitlich versetzt statt, da anscheinend einer der Prüflinge zuvor etwas zu spät kam. Meine Prüfung war für 11:00 angesetzt, angefangen habe ich ca 11:20. In der Prüfung waren nur Prof. Steinmann und Frau Keller anwesend - dabei saß Prof. Steinnmann gegenüber und Frau Keller seitlich neben mir. Während der Prüfung hat Frau Keller protokolliert.

Prof. Steinmann fragte zu Beginn der Prüfung, mit welchem Thema ich beginnen möchte (wie aus den anderen Protokollen hervorgeht). Ich wählte Interfaces und die interface basierte Programmierung. Dazu erklärte ich zunächst, was ein Interface ist: Es legt fest was implementiert wird, aber nicht wie. Dazu werden die Methodennamen und -signaturen deklariert, aber keine Methodenrümpfe - diese kommen in die implementierenden Klasse. Die interfacebasierte Programmierung erkennt man an der Verwendung von Interfaces, der Implementierung von Interfaces und der Deklaration von Variablen als Typ eines Interfaces. Davon ausgehend habe ich die folgenden Punkte ausführlich beschrieben: Polymorphismus und Entkopplung des Systems durch Verwendung von Interfaces als Typen, strukturelle und nominale Typkonformität, Interfaces vs abstrakte Klassen und fragile base class problem, Aufteilung von Interfaces in totale und partielle Interfaces und öffentliche und veröffentlichte Interfaces. Danach bin ich auf die Klassifizierung von Interfaces gekommen, dazu erst die Betrachtung von Aufruferin und Aufgerufene erläutert. Die Klassifizierung habe ich von der Allgemeinheit her, also allgemeine und kontextspezifische Interfaces erklärt und vom Nutzen, also anbietende und ermöglichende Interfaces. Danach 5 Beispiele dafür: Als anbietende, allgemeine Interfaces idiosynkratische und Familien-Interfaces, als anbietendes, kontextspezifisches das Client/Server Interface und als kontextspezifische, ermöglichende Server/Client und Server/Item Interfaces. Darauffolgend habe ich die anfangs erwähnte Entkoppelung wieder aufgegriffen und gesagt, dass die Typisierung der Variable durch das Interface der erste Schritt ist, aber danach der Variable auch noch ein Objekt zugewiesen werden muss. Das Interface und das Objekt müssen typkonform sein - aber wenn die Klasse explizit genannt wird, besteht wieder eine Abhängigkeit. Dem kann durch Dependency Injection entgegengewirkt werden - also habe ich Konstruktor, Setter und Interface Injection erklärt. Danach habe ich durch das Interface ermöglichte DIP erklärt und dafür ein Beispiel mit Paketen in Java gegeben. Abschließend habe ich erläutert, das hinter dem Interface eine bestimmte Entwurfsentscheidung "versteckt" werden soll. Was dem Interface allerdings fehlt, ist die Erzwingung der Einhaltung einer bestimmten Spezifikation, einem Vertrag. In dieser Hinsicht kann DbC und Unit Tests verwendet werden, um die Einhaltung der Spezfikation zu ermöglichen.

An dieser Stelle habe ich Prof Steinmann gefragt, ob ich darüber DbC und Unit-Tests zusammenfassen kann. Er hat zugestimmt. Ich habe kurz die Vor- und Nachbedingungen mit dem im Skript gebrachten Beispiel der Kundin und dem Flug gebracht. Dann bin ich direkt zu der Spezfikationssprache übergegangen, habe JML erläutert und dazu auch Eiffel (ensure, require und old erläutert) und den Unterschied: In JML können Modellmethoden und -variablen verwendet werden, wodurch die Methoden eines Interfaces komplett spezifiziert werden können und somit die implementierenden Klassen diese Spezifikation einhalten müssen.

Danach bin ich zu den Unit-Tests übergangen und habe erläutert, dass ein verified design by contract noch nicht möglich ist und somit immer nur ein Pfad im Programm gerpüft wird. Ein Testfall definiert eine Eingabe, erwartete Ausgabe und tatsächliche Ausgabe des Programms und vergleicht die Ausgaben - unterscheiden sich die Ausgaben, dann schlägt der Test fehl. Dafür kann aber für jede Methode mehrere Testfälle definiert werden, wodurch jeder mögliche Pfad abgedeckt werden kann. Somit können auch für die Interfaces für jede Methode Testfälle angegeben werden, wobei das Problem ist, dass von dem Interface kein Objekt instanziiert werden kann, auf welchem sich die Tests ausführen lassen. Allerdings kann für ein Interface eine abstrakte Testklasse angegeben werden, wobei das Objekt, auf welchem getestet werden soll, durch Factory-Methode zugewiesen wird. Das tatsächliche Objekt hängt somit von der Klasse ab, welche das Interface implementiert. Abschließend sagte ich, dass die Unit-Tests und die Spezifikationssprache des DbC der Einhaltung der Spezifikation von Interfaces dienen.

Ich habe zwar keine Uhr dabei gehabt, aber nach dem Vortrag waren locker 20 Minuten vorbei.

Prof Steinnmann fragte an dieser Stelle, was ich zum Java Schlüsselwort "assert" weiß und was die Probleme sind.
Meine Antwort: Assert wird in Java für das Prüfen von Zusicherung verwendet, wobei bei assert nicht auf alte Variablenwerte zugreifen kann. Ein weiterer Nachteil ist, dass assert den Aufruf von nebeneffektvollen Methoden gestattet. Dazu habe ich erklärt, was genau nebeneffektvolle Methoden sind - und habe noch eine Überleitung zu JML geschaffen, dass dort der Aufruf von nebeneffektvollen Methoden durch das pure Statement verhindert werden soll. Das kann aber auch fehleranfällig sein, da die Programmiererin die Methoden selbst als pure kennzeichnet und niemand sie daran hindert, eine nebeneffektvolle Methode mit pure zu kennzeichnen. Darüber hinaus habe ich das Problem von nebeneffektvollen Methoden im Testbetrieb, wenn assert aktiviert ist und im Livebetrieb, wenn assert deaktiviert ist, erläutert.

Prof Steinmann fragte mich daraufhin, was mir zu Refactoring einfällt.
Meine Antwort: Refactorings werden dazu verwendet, den Code zu verbessern, ohne das beobachtete Verhalten zu ändern. Als Beispiele für Refactorings habe ich solche genannt, die die Lesbarkeit verbessern, Daten organisieren, Generalisierung einsetzen, Methoden organisieren und Bedingungen vereinfachen.

Prof Steinmann fragte, was ein Metaprogramm ist:
Hier sagte ich, das ein Metaprogramm ein anderen Programm liest und ändert. Das war falsch, ein Metaprogramm hat sich selbst zur Ein und Ausgabe und kann sich dabei selbst verändern. Ich bin dazu übergangen, dass Metaprogrammierung durch Reflection möglich ist und habe die Teilbereiche Introspektion, Interzession und Modifikation erläutert.

Prof Steinmann fragte, ob ein Refactoring Tool ein Metaprogramm ist. Eventuell wollte er mich dadurch meinen Denkfehler der vorhergehenden Aufgabe erkennen lassen. Ich bin hier leider auf keinen grünen Zweig gekommen.

Prof Steinmann fragte, was die Probleme der Metaprogrammierung sind. Hier habe ich erläutert, dass ein Metaprogramm, welches sich selbst verändert, nicht mehr das macht, wofür es erstellt wurde. Die richtige Antwort wäre gewesen: Die Lesbarkeit und Wartbarkeit verschlecherten sich bei einem Programm, welches sich selbst verändert.

Damit war die Prüfung abgeschlossen. Auch wenn ich bei den letzten Fragen nicht wirklich brillieren konnte, war ich mit meinem Ergebnis mehr als zufrieden.

Prüfungsprotokoll

20. September 2012

Kurs 01853 – Moderne Programmiertechniken und -methoden

Lehrgebiet Programmiersysteme

Prof. Dr. Friedrich Steimann

Prof. Steimann beginnt mit der Frage, mit welchem Themengebiet ich anfangen möchte.

Ich erläuterte daraufhin die wichtigsten Konzepte der interfacebasierten Programmierung (Was ist ein Interface?, Interfaces als Typen, Zusammenhang Interfaces und abstrakte Klassen, totale und partielle Interfaces, Klassifizierung von Interfaces, Dependency Injecction, Interface Segregation Principle, Dependency Inversion Principle, Interpretation von Interfaces als Rollen).

Dann bin ich nach Zustimmung von Prof. Steimann auf das Themengebiet Design by Contract übergegangen und erklärte hier die bedeutendsten Aspekte (Definition Design by Contract, Vor- und Nachbedingungen, Klasseninvarianten, Zeitpunkt der Überprüfung der Zusicherungen, Beeinflussung der Progranmierung, Zusicherungen und Substituierbarkeit, Design by Contract als Form des Testens, Verified Design by Contract, Vor- und Nachteile des Design by Contract).

Abschließend hat Prof. Steimann mich auf die Metaprogrammierung angesprochen. Hier erläuterte ich die Verwendungen und Voraussetzungen der Metaprogrammierungen und ging auf das Thema der Reflektion mit Introspektion, Interzession und Modifikation sowie auf Annotationen und Attribute ein.

Prof. Steimann hatte keine Zwischenfrage.

Die Prüfung wurde mit 1,0 bewertet.

Prof. Steimann ist ein sehr freundlicher und aufgeschlossener Prüfer und möchte den Studenten keineswegs „auf die falsche Fährte“ führen. Ich bin mit dem Ablauf der Prüfung vollkommen zufrieden und empfehle den Kurs 01853 und das Lehrgebiet Programmiersysteme uneingeschränkt weiter

Prüfungsprotokoll 1853

Kurs:	01853 „Moderne Programmiertechniken und -methoden“
Prüfer:	Prof. Dr. Friedrich Steimann
Beisitzerin:	Dr. Daniela Keller
Prüfungstag:	29. Juni 2012
Note:	Sehr gut

Situation

Etwa 20 Minuten vor meinem Prüfungstermin traf ich ein und meldete mich bei Frau Schmidt an. Anschließend wartete ich etwa eine halbe Stunde im Zwischenflur. Frau Keller rief mich dann herein. Ich legte ihr meinen Studentenausweis und meinen Personalausweis vor und sie überreichte mir die Bescheinigung für das Finanzamt. Prof. Steimann saß mit etwas Abstand am anderen Ende des Schreibtischs und eröffnete die Prüfung mit folgender Frage:

Prüfungsfragen

Womit möchten Sie beginnen?

⟶ Interfacebasierte Programmierung

Dann mal los!

⟶ Entwurfsentscheidungen hinter Schnittstelle kapseln, reduzierter Änderungsaufwand, in OOP Klassen als Module, Klasseninterface, C# und Java bieten zusätzlich Interfaces. Öffentliches Interface, veröffentlichtes Interface, nicht änderbar. Interface-Implementierung, explizite Interfaces in C#, nominale Typkonformität. Totale Interfaces, Partielle Interfaces, Interface Segregation Principle, allgemeine Interfaces, ideosynkratisches Interface, Familien Interface, anbietende Interfaces, Client/Server-Interface, kontextspezifisches Interface, ermöglichende Interfaces, Server/Client-Interface, Server/Item-Interface. Abhängigkeit durch Instanziierung, Dependency Injection per Konstruktor, Setter oder Interface, Assembler und Factories, Interfaces bieten keine Semantik, Lösung per Design-by-Contract oder Unit Tests. (etwa 10 Minuten)

Sie haben die Brücke ja schon gegeben. Ich interessiere mich aber mehr für Design-by-Contract. Was können Sie dazu sagen?

⟶ Verträge explizit machen. Vorbedingungen, Nachbedingungen und Invarianten. Zeitpunkte der Überprüfung. Problem mit Aliasing bei Invarianten.

Wie kann man das Verhalten von Interfaces spezifizieren?

⟶ Mit den Modellvariablen und Modellmethoden in JML. Per Abstraktionsmethoden dann auf konkrete Klasse abbilden. Alternativ mit Unit-Tests.

Was müssen Design-by-Contract-Sprachen mindestens mitbringen und wie unterscheiden sich die Sprachen Eiffel, Java und JML?

⟶ Sprachen müssen Zugriff auf alten Zustand ermöglichen. Eiffel: Methoden mit Seiteneffekt sind zwar nicht erlaubt, wird vom Compiler aber nicht sichergestellt. Java: Asserts sind ebenfalls nicht nebeneffektfrei; Seiteneffekte müssen sogar benutzt werden um alten Zustand zu speichern. JML ist einzig sichere Sprache wegen `@pure`-Annotation; fehlerhafter Vergabe der Annotation aber möglich.

Ok. Und was wissen Sie über Meta-Programmierung?

⟶ Definition. Introspektion, Interzession, Modifikation. Annotationen. Aspektorientierte Programmierung: Motivation (Crosscutting concerns, Scattering, Tangling).

Nennen Sie jeweils die Probleme der drei Ausprägungen.

⟶ Schreibweise bei Annotationen schlechter als bei Marker-Interfaces. Man sieht dem Code nicht an, dass Aspekte auf ihn wirken. Aspekte können versehentlich auf Stellen angewandt werden, die eigentlich nicht gewünscht sind. Zu Nachteil der Modifikation viel mir nichts ein, habe dann das Vorgehen in Java erklärt (via Betriebssystem und Class-Loader). *(Prof. Steimann: Code kann bei Modifikation nur zur Laufzeit betrachtet werden, undenkbar wenn es um Zertifizierung oder Ähnliches geht.)*

Warum ist das Abfrage von Annotationen aufwendiger als das Abfragen von Marker-Interfaces?

⟶ Hier hatte ich etwas gebraucht um auf den Punkt zu kommen. Subtyping fehlt. Es können (weil kein Compiler verwendet wird) viele Fehler auftreten, die abgefangen werden müssen. *(Prof. Steimann: Abfrage findet vollständig zur Laufzeit statt, kein `instanceof`-Operator)*

Welcher Zusammenhang besteht zwischen Annotationen und Reflektion?

⟶ Annotationen können einen besseren Einstieg in das Programm bieten weil die Stellen im Code sichtbar werden, die von Reflektion betroffen sind. Beispiel: JUnit's `@Test` -Annotation.

Ende

Nach der letzten Frage bat mich Prof. Steimann draußen zu warten. Nach kurzer Wartezeit rief er mich dann wieder herein. Er erläuterte kurz meine Note und erkundigte sich dann nach meinen weiteren Studienplänen an der FernUni.

Die Themen Unit-Testen, Entwurfsmuster und Extreme Programming waren nicht Teil der Prüfung (obwohl ich öfter versuchte auf Unit-Testen überzuleiten). Die Prüfung verlief insgesamt ruhig ab. Herr Steimann ließ mir viel Zeit für die Beantwortung der Fragen und unterbrach mich nie. Ich wusste allerdings meistens auch nicht, ob meine Antwort dem entsprach, was er hören wollte.

Prüfung und Kurs sind meiner Meinung nach sehr empfehlenswert. Die Themen sind für mich als Softwareentwickler sehr interessant und ich konnte beruflich bereits einiges aus dem Kurs anwenden. Um inhaltlich viel mitzunehmen, lohnt es sich vorher den Kurs 01814 „Objektorientierte Programmierung“ zu bearbeiten.