Prüfungsprotokoll Praktische Informatik
Datum: 5.7.2000
Kurs: 1814 Objektorientierte Programmierung
Prüfer: Prof. Dr. A. Poetzsch-Heffter
Dauer 20-25 Min.
Note 3,0

Fragen:
Geben Sie einen Überblick über die Struktur des Kurses.
Was ist Subtyping? Gibt es Subtyping auch ohne Vererbung? Wo kommt das vor und wofür braucht man das? (=> Schnittstellen).
AWT: Wo spielen Schnittstellen im AWT eine Rolle? Welche 3 Hauptaspekte kennzeichnen das AWT? Wie funktioniert ein ActionListener? Welche Methoden muß man selbst implementieren? Wenn er bei einem Button eingetragen wird: was passiert genau bei einem Mausklick, wer ruft welche Methoden bei welchen Objekten auf?
Verteilte Programmierung: Wie funktioniert das grundsätzlich? Was ist der Unterschied zwischen RMI und Sockets? Welche Vor- und Nachteile ergeben sich jeweils? (=> Laufzeitfehler, Effizienz); Wie funktioniert RMI? Woher weiß ein Prozeß welche Objekte es in anderen Prozessen gibt? Wie bekommt man eine Referenz auf diese Objekte? Wie werden die Methoden aufgerufen?

Die Prüfungsatmosphäre war freundlich und entspannt, obwohl ich öfter nicht weiterwußte. Wenn etwas unklar blieb wurde allerdings schon nachgefragt um den Sachverhalt zu präzisieren. Die Benotung mit 3,0 kam mir wohlwollend vor. Ich hatte mir auch insgesamt nur relativ wenig Zeit genommen, um den Kurs durchzuarbeiten und mich auf die Prüfung vorzubereiten. Nach Aussage von Prof. Poetzsch-Heffter kommen durch die Bank alle Noten von 1,0 bis 5,0 vor.

Gedächtnisprotokoll zur Diplomprüfung Kurs 1814 Objektorientierte Programmierung

Datum: 09.06.2000
Prüfer: Prof. DR. Poetzsch-Heffter
Beisitzer: Herr Müller
Dauer: 25 min
Kursversion: SS 99

- Überblick über den Kurs
- Was ist eine Klasse?
- Unterschied zuwischen Klasse und Interface
- Abstrakte Klassen (allg. Erklären, besonders: Kanu eine abstrakte Klasse von einer "normalen" Klasse abgeleitet sein? (Ja!))
- Dynamische Methodenauswahl
- Beispiele für statisches Linken (Attribute, Methoden mit "static")
- Was ist mit dem Aufruf super.methode() ? (statisch gelinkt, bei dynamischem Linken könnte es u.U. dazu kommen, dass die Methode sich immer wieder selber aufruft)
- Verteiltes Programmieren (Unterschiede zu sequentiellen Programmen / Stub-Skeleton-Mechanismus / Möglichkelten, Parameter und Rückgabewerte zu behandeln)

Ich kann nur bestätigen, dass Prof. Poetzsch-Hefftter als Prüfer zu empfehlen ist. Die Prüfung verlief in sehr angenehmer, wenn auch fordernder Atmosphäre. Ich hatte immer genug Zeit zum Nachdenken und konnte mir Sachen auch herleiten, wenn ich sie nicht auswendig wusste.

Prüfung: Objektorientierte Programmierung, Kurs 18 14

Prüfer: Prof. Poetzsch-Heffter

Beisitzer: Dr. Müller

Datum: März 1999

1. Inhaltsangabe zum Kurs
2. Subtyping: allgemein darstellen
3. Vererbung: erstmal allgemein, dann
    1. Eine Klasse B wird von einer Klasse A abgeleitet, in B wird eine Instanzvariable von A überschrieben. Existieren diese in B dann beide? Antwort: ja.
    2. Wie sieht man, daß das so sein muß?
    3. Anders gefragt: An welchem Beispiel sieht man, daß das Gegenteil keinen Sinn macht?
    4. Beispiel angeben für den Fall, daß dynamische Methodenauswahl nicht durch statische Bindung ersetz werden kann. Antwort: Wie im Kurs im Beispiel mit den "druckbar"-Objekten: einer Variablen mit dem statischen Typ der Superklasse werden etwa in einer Schleife nacheinander Objekte verschiedener Subklassen zugewiesen, die alle ihre eigene Form der Methode “druckbar” implernentieren.
4. AWT: zunächst die Hierarchie skizzieren
    1. Warum ist es günstig, so eine Hierarchie zu haben, im Gegensatz zu einer Deklarierung der einzelnen Klassen, ohne sie voneinander abzuleiten?
    Antworten: dynamische Methodenauswahl ist dann anwendbar; das AWT ist leichter für eigene Zwecke erweiterbar; eine Methode kann den allgemeinsten Typ, Component, als Parameter haben und alle Subtypen sind dann auch dort einsetzbar.
5. Threads beschreiben, dann
    1. Welche Variablen benutzen sie gemeinsam, welche hat jeder thread selbst? (Stichworte: Adreßbereich, Instanzvariable, Stack)
    2. Wie werden die Methoden wait(), notify(), notifyAll() benutzt?

Die Prüfungsatmosphäre war sehr angenehm und fordernd.

Kurs 01814 Objektorientierte Programmierung (04/97)
mündliche Diplomprüfung, Dauer 25 min
Prüfer: Prof. Dr. Poetzsch-Heffter

Foto vom Prüfer: http://www2.informatik.tu-muenchen.de/pics/Arnd.gif

Eindruck vom Prüfer: nett, versucht Nervosität zu nehmen.

Welche Vorzüge bietet das objektorientierte Konzept gegenüber prozeduralen Programmiersprachen?
Wozu benötigt man das dynamische Binden? - Schreiben Sie ein Beispiel auf!
Was ist ein Thread?
Welche Probleme treten auf?
1: Synchronisation - wie und wozu
2: Monitorkonzept
(Monitor sitzt beim Objekt)
Wie funktioniert Garbage collection (nicht im Kurs enthalten)
Aber: Vor- und Nachteile automatischer Garbage collection
(Weniger Programmierarbeit, aber bei Ressourcenknappheit keine Kontrolle über Speicherfreigabe)
Nennen Sie ein Beispiel: Wozu braucht man Multithreading?
(Z.B: HTML-Seite einlesen, quasi-parallel Bilddateien einlesen und Applets ausführen)

Diplomprüfung
Praktische Informatik

Prüfer: Prof. Poetzsch-Heffter
Beisitzer: Hr. Müller
Termin: 03.09.97
Dauer: 30 min
Note: 1,0
Prüfungsinhalte: 1814 Objektorientierte Programmierung

Prüfungseindruck:
Prof. Poetzsch-Heffter ist ein sehr fairer und angenehmer Prüfer, der ruhig und klar sein Fragen stellt. Er hat meines Erachtens einen ähnlichen Fragestil wie die übrigen Professoren. Er stellt allgemeine Fragen, wie etwa "Was versteht man unter ..." kann jedoch auch "punktgenauer" nachfragen. Zur Prüfung hatte er sich die Fragen notiert (ca. 60), ist diese aber nicht der Reihe nach durchgegangen, sondern die Liste dient nur zur Übersicht.
Man muss die prinzipiellen Konzepte der OO-Programmierung kennen. Es kommen Transferfragen, wie z.B. "Warum ist OO zur GUI-Programierung hilfreich?". Java ist im Detail nicht wichtig! Der Hauptaugenmerk liegt auf der allg. OOP, man sollte aber natürlich die generelle Umsetzung in Java kennen, wie z.B. Subtyping mit mehreren Supertypen trotz der "fehlenden" Mehrfachvererbung.
Ein globales Verständnis, insbesondere der Zusammenhänge, ist genauso wichtig wie Details der einzelnen zentralen Konzepte zu kennen!
Insgesamt ist es eine einfache Prüfung, wenn man die OO-Denkweise verinnerlicht hat.

Prüfungsverlauf:
1. Was versteht man unter Spezialisierung?
2. Subtyping, Subclassing erläutern.
3. Überladen von Methoden erläutern; allgemein in der OOP und die spezielle Situation (Unterschiede) in Java.
4. Wie müssen bei überladenen Methoden die Typen der Parameter und Rückgabewerte in Beziehung stehen.
    - Genau angeben was Super-und was Subtyp sein muß. Am besten erklären (herleiten) warum dies so sein muß!
5. Können Sie die Gründe nennen, warum die OOP zur GUI-Programmierung besonders geeignet ist?
    - Methodische Gründe und die praktische SW-Konstruktion mittels Vererbung vom GUI-Toolkit (bei Java das AWT) nennen!
    - Idee der aktiven Objekte (unterstützt durch Threads)!
6. Wie ist das AWT strukturiert?
    - Die Vererbungshierarchie zur Nutzung für die praktische SW-Konstruktion erläutern (Component ... Frame ... Button)
    - Eventbehandlung allgemein beschreiben.
7. Fehlerbehandlung in Java erläutern und anhand eines Beispiels die Komponenten:
    - try, catch, finaly, sowie
    - throws, erläutern und die Verbreitung von Exeptions beschreiben.
8. Wie erkennt der Compiler daß alle möglichen Exeptions abgefangen werden, um die Vollständigkeit der throws-Klausel zu prüfen?
9. Erläutern Sie das Client/Server Modell. Was ist der Vorteil gegenüber einer symetrischen Architektur?