Bewertungskriterien Klausur - Erreichte Lehrziele

Prof. Dr.-Ing. Thomas Schwotzer

1 Programmieren 2 - Lehrziel

Der Kurs Programmieren 2 dient der Vertiefung von Fähigkeiten und Kenntnissen in der objektorientierten Programmierung am Beispiel von Java. Die grundlegenden Lehrziele bestehen darin:

- Sie können aktiv Methoden der Vererbung und Polymorphie einsetzen.
- Sie können Interfaces aktiv einsetzen. Sie verstehen den Einsatz des Interfaces als Kontrakt.
- Sie kennen das Konzept der testgetriebenen Entwicklung und sind in der Lage, die Lauffähigkeit Ihres Programmes mittels JUnit-Tests zu beweisen. Sie kennen das Prinzip der *Facade*.
- Sie nutzen aktiv die von Java bereit gestellten Strukturen zum Speichern von Daten (wie `Set, List, Map`.
- Sie verstehen das Konzept der Streams in Java und den Zusammenhang zum unterliegenden Betriebssystem.
- Sie können einfache UML-Klassendiagramme schreiben.
- Sie bauen Ihre Programm nach den OO-Prinzip auf nicht nach Prinzipien prozeduraler Entwicklung. Ihr Programm ist primär ein Netzwerk kommunizierender Obkjekte und nicht nur eine Ansammlung von Methoden.
- Sie haben die grundsätzlichen Prinzipien der Java-Programmierung aus dem Kurs Programmieren 1 nicht vergessen und nutzen diese aktiv.

2 Bewertungskriterien

Diese Lehrziele werden im Rahmen der Programmierklausur getestet. Sie implementieren ein einfaches objektorientiertes Programm. Bewertet wird, ob die oben genannten Lehrziele erreicht wurden. Die Bewertungskriterien sind diese:

Lauffähigkeit 20% Das Programm läuft und erfüllt die Aufgabenstellung vollständig.

Zweckmäßigkeit 20% Datentypen und Sprachelemente von Java wurden zweckmäßig genutzt. Beispiel für Unzweckmäßigkeit:

- Vererbung wurde nicht eingesetzt, obwohl sie zweckmäßig gewesen wäre.
- Offensichtlicher Verstoß gegen das Prinzip *program to interface*.
-

Das sind nur einige Beispiele.

Stil 20% Programme müssen wartbar sein. Sie müssen einfach verstanden werden können. Daher sind die formalen und informellen Richtlinien von Java zu beachten. Das betrifft das Einschieben, Dokumentation von Methoden (insbesondere in Interfaces) die Benennung von Variablen, Methoden usw.

Generell ist Stil alles, was die Lesbarkeit eines Programmes beeinflusst.

Interfaces / Exceptions 20% Interfaces sind ein Grundbaustein professioneller OO-Software. Sie nutzen Interfaces zweckmäßig. Sie nutzen Interface-Vererbung, wenn es geeignet ist. Sie erkennen und nutzen passende Exceptions.

Tests 20% Sie beweisen die Lauffähigkeit Ihres Programmes durch ausreichende JUnit-Tests. Sie testen nicht nur *Gut*-Fälle, sondern auch die Reaktion auf fehlerhafte Eingaben oder Randfälle.

Die Programmierklausur wird unter Einsatz einer IDE stattfinden. IDE helfen enorm gut bei der Erstellung eines Programmes. Daher gilt auch:

Ein Programm, das nicht kompiliert wird als ungenügend (5,0) bewertet. Das sollte wirklich nicht schrecken, denn die IDE machen es nahezu unmöglich, Kompilierfehler nicht zu erkennen.