HSLU Hochschule Luzern

Objektorientierte Programmierung (OOP)

MEP OOP - FS24: Information und Stoffabgrenzung

Roland Gisler, Version 1.0.0 (FS24)

Organisation:

Datum:	**Dienstag**, **18. Juni** 2024, **08:30** – **10:30** Uhr
Ort/Raum:	siehe InfoScreens (vor Ort in Rotkreuz)
Dauer:	**105** Minuten (reine Prüfungszeit)
Prüfung:	Elektronische Programmierprüfung auf Ihrem Rechner, mit individualisierten Aufgaben. Abgabe über ILIAS.
Durchführung:	Für die Prüfung benötigen Sie einen Rechner mit Netzwerkzugang, Entwicklungsumgebung (Java, Maven, IDE) sowie Zugang zu ILIAS.

Rahmenbedingungen:

- Die Prüfung findet «open book» statt, d.h. der Gebrauch der Unterlagen ist gestattet. Beachten Sie aber, dass die Zeit für umfangreiche Recherchen nicht ausreichen wird. Sie müssen sich also gut vorbereiten, und die Grundlagen präsent haben.
- Die Prüfung ist persönlich zu bearbeiten. Insbesondere verboten sind die Inanspruchnahme von Unterstützung und Kommunikation mit Dritten während der Prüfung, sowie die Weitergabe von Prüfungsaufgaben oder -lösungen in jeder Form.
- Allfällig festgestellte unlautere Vorkommnisse haben umgehend den Abbruch und das Nichtbestehen der Prüfung zur Folge, sowie allfällige weitere Konsequenzen. Es gilt Artikel 39 der Studienordnung für die Ausbildung an der Hochschule Luzern, FH Zentralschweiz (Nr. 521).

Stoffumfang

Mit der MEP wird die Erreichung der Lernziele des Moduls überprüft. Das umfasst den Stoff **aller Inputs**, der **Übungen** und auch die Kapitel aus dem **Buch**. Zur einfacheren Orientierung finden Sie folgend die zusammengefassten Lernziele aus allen **Inputs** des Modules.

Objektorientierung (O01- O17):

- Sie haben eine erste Vorstellung von der Objektorientierung.
- Sie können die Begriffe Klasse, Objekt, Zustand und Verhalten erklären und einordnen.
- Sie kennen und verstehen die zentralen "Bestandteile" einer Klasse.
- Sie kennen die Merkmale der verschiedenen Arten von Variablen.
- Sie können zwischen der Methodenimplementation und dem Methodenaufruf, sowie zwischen formalen und aktuellen Parametern unterscheiden.
- Sie sind in der Lage eine einfache Klasse schemamässig zu implementieren.
- Sie kennen die elementaren Datentypen `boolean`, `byte`, `short`, `int`, `long`, `float`, `double` und `char`.
- Sie sind sich dem Unterschied zwischen Wertebereich und Genauigkeit bewusst.
- Sie kennen die von Datentypen abhängige Wirkung von Operatoren.
- Sie können einfache Typumwandlungen zwischen den Datentypen durchführen.

- Sie können einfache Bedingungen und boolsche Ausdrücke formulieren.
- Sie kennen das `if`-, das `else if`-, das `switch`-Statement und die `switch`-Expression.
- Sie kennen die Vor- und Nachteile der verschiedenen Selektions-Statements und können gezielt das jeweils am besten geeignete Statement auswählen.
- Sie kennen die verschiedenen Schleifen-Typen von Java (`while`, `do-while` und `for`) mit ihren individuellen Eigenschaften.
- Sie können für jede Situation den geeigneten Schleifentyp auswählen und implementieren.
- Sie sind in der Lage, sichere Abbruchbedingungen zu Formulieren.
- Sie beherrschen den Entwurf von abstrakten Klassen.
- Sie sind in der Lage Schnittstellen (Interfaces) zu definieren.
- Sie können Schnittstellen einsetzen und implementieren.
- Sie können für ausgewählte Elemente eine Javadoc schreiben.
- Sie verstehen das Konzept der Vererbung.
- Sie können zwischen der Vererbung von Interfaces und Klassen unterscheiden.
- Sie kennen den Unterschied zwischen Einfach- und Mehrfachvererbung.
- Sie sind in der Lage gute und schlechte Vererbungen zu identifizieren.
- Sie wissen wie man Methoden in Spezialisierungen überschreiben, und die Implementation der Basisklasse wiederverwenden kann.
- Sie können den Begriff Polymorphismus anhand von Beispielen erklären.
- Sie kennen die Technik des Überladens und des Überschreibens von Methoden.
- Sie sind mit dem Konzept des Subtyping vertraut.
- Sie können zwischen statischem und dynamischem Typ unterscheiden.
- Sie können generisch implementierte Klassen nutzen.
- Sie kennen die unterschiedlichen Arten von «Gleichheit».
- Sie verstehen den Inhalt und die Notwendigkeit des equals- und hashcode-Contracts.
- Sie sind in der Lage adäquate Implementationen von `equals()` und `hashcode()` vorzunehmen, zu beurteilen und zu testen.
- Sie können die natürliche Ordnung mit `Comparable` festlegen.
- Sie können einfache Comparatoren für spezielle Ordnungen implementieren.
- Sie kennen den Sinn und Zweck von mit `static` als statisch ausgezeichneten Attributen und Methoden.
- Sie können das Schlüsselwort `final` in den verschiedenen Kontexten von Variablen, Methoden und Klassen sinnvoll anwenden.
- Sie können Enumerationen definieren und anwenden.
- Sie kennen verschiedene Techniken zur Fehlerbehandlung.
- Sie können verschiedene Fehler(-situationen) abhängig vom Kontext beurteilen.
- Sie kennen das Konzept der Ausnahmebehandlung.
- Sie können in Java Exceptions definieren, behandeln und selbst auslösen.
- Sie wissen, wie man Ausnahmen mit JUnit explizit testen kann.
- Sie verstehen das Observer- bzw. Event/Listener-Pattern in Java.
- Sie können ein Listener-Interface implementieren und nutzen.
- Sie können eigene Event-Typen und Eventquellen implementieren.
- Sie erkennen das Potenzial mit dem Event/Listener-Pattern, die Kopplung zwischen Softwareeinheiten massiv zu senken.
- Sie haben eine Vorstellung, was (anonyme) innere Klassen sind.
- Sie können einfache (anonyme) innere Klassen zur Event-Behandlung implementieren.
- Sie verstehen das Konzept der Datenströme.
- Sie können Byte- und Zeichen-Datenströmen unterscheiden und kennen die Unterschiede.
- Sie wissen, was Encoding ist, und sind sich der potenziellen Probleme bewusst.
- Sie können einfache Byte- und Datenströme in Dateien lesen und schreiben.
- Sie kennen einige fundamentale Klassen für das Dateihandling.
- Sie können den Begriff «Funktionale Programmierung» einordnen.
- Sie wissen, was eine Lambda-Expression ist.
- Sie wissen, was ein «Functional Interface» ist.
- Sie können in Java einfache Lambda-Expressions implementieren und anwenden.
- Sie kennen das Prinzip des «Stream»-Programming.

- Sie können einfache Beispiele die auf Streams und funktionaler Programmierung (Lambdas) basieren nachvollziehen.
- Sie kennen das Konzept der Supplier-, Intermediate- und Terminal-Operationen für/auf Streams.
- Sie können einfache Anforderungen mittels der Stream API umsetzen.
- Wie wissen was AutoBoxing ist und können damit adäquat umgehen.
- Sie kennen grundlegende Konzepte wie GUIs programmiert werden.
- Sie können die Herausforderungen der GUI-Programmierung anhand von Beispielen erläutern.
- Sie verstehen die Absicht des Model-View-Control (MVC) Prinzips.
- Sie wissen, welche Herausforderungen in der GUI-Programmierung durch die Plattformunabhängigkeit entstehen.
- Sie kennen die drei technologischen Ansätze für die GUI-Programmierung mit Java.
- Sie sind in der Lage, Code von GUI-Programmen zu verstehen und Änderungen vorzunehmen.

Datenstrukturen (D01 und D02):

- Sie wissen, was Datenstrukturen sind und wofür man sie einsetzt.
- Sie kennen verschiedene Eigenschaften von Datenstrukturen.
- Sie kennen den groben Umfang des Java Collections Frameworks.
- Sie kennen einige grundlegende Schnittstellen, Algorithmen und Implementationen des Java Collections Frameworks.
- Sie verstehen die Klasse `ArrayList` als konkrete Implementation einer `List` und können diese nutzen.
- Sie sind in der Lage, eine den Anforderungen angemessene Datenstruktur auszuwählen.
- Sie kennen einige ausgewählte Datenstrukturen und ihre Semantik.
- Sie verstehen das Konzept des Iterators und können diesen auf verschiedenen Datenstrukturen einsetzen.
- Sie können ausgewählte Datenstrukturen praktisch nutzen und einfache Operationen auf/mit diesen durchführen.

Prinzipien (P01 – P03):

- Sie verstehen den Vorgang der Abstraktion bei der OO-Modellierung.
- Sie verstehen die grundlegende Idee der Modularisierung.
- Sie beherrschen das Konzept der Datenkapselung.
- Sie kenne die Wirkung der Zugriffsmodifizierer in Java und können diese adäquat einsetzen.
- Sie verstehen die Designaspekte des Information Hiding.
- Sie kennen fundamentale Ziele beim objektorientierten Design.
- Sie können die Kohäsion einer Klasse beurteilen.
- Sie können die Kopplung zwischen Klassen beurteilen.
- Sie vermögen den Zielkonflikt zwischen hoher Kohäsion und loser Kopplung abzuwägen.
- Sie kennen zehn fundamentale Designprinzipien.
- Sie können diese Prinzipien in der Programmierung anwenden.
- Sie verstehen den Sinn, Zweck und Nutzen dieser Designprinzipen.

Entwicklungsprozess (E01 – E04)

- Sie wissen, was eine IDE ist und welchem Zweck sie dient.
- Sie kennen die wichtigsten Aufgaben einer IDE.
- Sie können mit einer IDE grundlegende Programmiertasks erledigen.
- Sie kennen eine für Java-Projekte adäquate Projekt- bzw. Verzeichnisstruktur.
- Sie kennen die Motivation, den Sinn und den Zweck des Testens.
- Sie wissen, was Sie mit Tests erreichen können und was nicht.
- Sie kennen verschiedene grundlegende Testarten und –verfahren.
- Sie können in Ihrer Entwicklungsumgebung einfache und gute Unit Tests, basierend auf dem JUnit-Framework, implementieren und anwenden.
- Sie kennen die Vorteile von Test First.
- Sie wissen, wie man das Fehlerhandling mit JUnit effizient testen kann.
- Sie kennen und nutzen zusätzliche Libraries wie (z.B. AssertJ) beim Testen.
- Sie kennen den Nutzen des Einsatzes von Logging.
- Sie können einfaches Logging (z.B. mit SLF4J und LogBack) nutzen.
- Sie sind in der Lage Fehlermeldungen zu lesen, verstehen und im jeweiligen Kontext einzuordnen.
- Sie kennen und nutzen, wenn sinnvoll, AssertJ beim Testen.
- Sie können Lambdas in (JUnit-)Tests und beim Logging nutzen.
- Sie sind in der Lage einfache Programme zu Debuggen.

Roland Gisler, 23. Mai 2024