Programmierung II

Allgemeine Angaben

Kürzel	PRG 2
Modulverantwortliche	Prof. Dr. Seifert, Prof. Dr. Ströder
Dozenten	Prof. Dr. Seifert, Prof. Dr. Hanheide, Prof. Dr. Ströder
Lehrsprache	Deutsch
Semester	2
ECTS-Punkte	5
Kontaktstunden	40
Selbststudium	85
Dauer	1 Semester
Art	Pflicht
Häufigkeit	jedes Studienjahr
Gewichtung	5/180
Prüfungsleistung	**K**RS90

Stichwörter

- Objektorientiertes Programmieren
- Spezifische Konstrukte objektorientierter Programmiersprachen
- Ausnahme- und Fehlerbehandlung
- Modellierung objektorientierter Software
- Clean Code

Zugangsvoraussetzungen

- Programmierung 1

Verwendbarkeit

- Alle Module der Informatik

Qualifikations- und Kompetenzziele

Die Studierenden kennen die Grundprinzipien des objektorientierten Programmierens. Sie verstehen und beherrschen die grundlegenden Konstrukte objektorientierter Programmiersprachen und können Programme auch höherer Komplexität zur Lösung konkreter Aufgaben und Probleme verstehen und selbständig entwickeln. Sie sind in der Lage, grundsätzliche Konzepte zur Modellierung objektorientierter Software anzuwenden. Sie haben außerdem einen exemplarischen Einblick in die Besonderheiten einzelner objektorientierter Programmiersprachen und deren Konstrukte gewonnen. Schließlich kennen sie grundsätzliche Prinzipien des Clean Code und können diese anwenden.

Lehr- und Lernmethoden

Unterschiedliche Lehr-/Lernumgebungen: Präsenzveranstaltungen, Eigenstudium; Wechselnde Lehr-/Lernmethoden: Individuelles und kooperatives Lernen, problemorientiertes und integratives Lernen anhand von Übungen.

Zur Vermittlung der Inhalte wird eine moderne und etablierte, objektorientierte und stark typisierte Programmiersprache eingesetzt, z.B. Java, C++ oder C#.

Inhalte

- Grundprinzipien des objektorientierten Programmierens
  - Klassen und Objekte, Attribute und Methoden
  - Kapselung und Information Hiding
  - Klassenhierarchie und Vererbung, Mehrfachvererbung, Kovarianz und Kontravarianz
  - Polymorphismus
- Spezifische Konstrukte objektorientierter Programmiersprachen
  - Namespaces
  - Interfaces
  - Bibliotheken und Frameworks
  - Ausnahme- und Fehlerbehandlung
  - Fortgeschrittene Datentypen (z.B. Container, Iteratoren)
  - Fortgeschrittene Sprachkonzepte (z.B. Attribute, Templates/Generics, Lambda-Ausdrücke)
- Modellierung objektorientierter Software und Prinzipien des Clean Code
  - Namenskonventionen und Coding Guidelines
  - Entwurfsprinzipien, Kopplung und Kohäsion, SOLID-Prinzip
  - Technische Schulden und Continuous Inspection
  - Modellierung mit UML (nur UML-Klassendiagramme und einfache Beispiele)
  - Entwurfsmuster (nur Konzept und einfache Beispiele wie Singleton oder Strategy)

Grundlegende Literaturhinweise

Je nach eingesetzter Programmiersprache, z.B.:

ABTS, D., 2020 : Grundkurs Java, Von den Grundlagen bis zu Datenbank- und Netzanwendungen, 11., aktualisierte und erweiterte Auflage, Springer Vieweg

ALBAHARI, J. und B. ALBAHARI, 2018. *C# 7.0. kurz & gut*. 5. Aufl. Heidelberg: O'Reilly. O'Reillys Taschenbibliothek.

LOUIS, D., 2018. *C++*. München: Carl Hanser Verlag.

THEIS, T., 2018. *Einstieg in C# mit Visual Studio 2017. Ideal für Programmiereinsteiger*. 5. Aufl. Bonn: Rheinwerk Verlag GmbH. Rheinwerk Computing.

Ergänzende Literatur

GAMMA, E., R. HELM, R. JOHNSON und J. VLISSIDES, 2015. *Design Patterns. Entwurfsmuster als Elemente wiederverwendbarer objektorientierter Software*. Frechen: mitp.

OESTEREICH, B. und A. SCHEITHAUER, 2013. *Analyse und Design mit der UML 2.5. Objektorientierte Softwareentwicklung*. München: Oldenbourg.