Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig

Mitglied der SUPSI

Objektorientierte Programmierung-5

Code	OOP_5 und OOP-P			
Fachbereich	Software-Entwicklung			
Studiengänge	BSc Informatik, MAS Informatik			
Art des Studiengangs	☒ Bachelor	☐ Master	☒ CAS/MAS/EMBA	
Studienniveau *	☒ Basic	☐ Intermediate	☐ Advanced	☐ Specialized
Typus **	☒ Core course	☐ Related course	☐ Minor course	
ECTS-Credits	5			
Präsenzverpflichtung	100% Das Praktikum (OOP-P) ist obligatorisch für Studierende ohne Programmiererfahrung.			
Arbeitsaufwand in Std.	150			
Verantwortliche Ansprechperson	Fachbereichsleiter: Peter Böhnlein	Autor: Corsin Capol		
Zu entwickelnde Kompetenzen	Die Studierenden sind mit den Konzepten und Techniken der Software-Entwicklung gemäss objektorientierter Programmierung (OOP) vertraut. Sie sind in der Lage, Problemstellungen zu analysieren, objektorientierte Lösungen zu entwerfen und in Java-Programme umzusetzen. Sie können die Regeln befolgen, die für eine qualitativ hochstehende und gut wartbare Programmstruktur massgeblich sind. Sie können Klassenbibliotheken zweckmässig in ihre Lösungen integrieren.			
Lerninhalte	Grundlegende Konzepte und Techniken der Software-Entwicklung gemäss OOP: • Objekte und Klassen, Klassenentwurf • Objektinteraktion / -kooperation • Delegation und Vererbung • Qualitätsregeln, Wartbarkeit • Klassenbibliotheken • Fehlervermeidung und -handhabung, Test-Strategien • Dokumentation			
Lehr- und Lernmethoden (Fernstudium nach dem Blended-Learning-Konzept)	Selbststudium • Erarbeiten des Stoffes • Lektüre • Lösen von Aufgaben • Lösen von Fallstudien, etc.	Online-Studium • Forumsdiskussionen • Einreichen von Aufgaben • Repetitionsaufgaben • Online-Feedback, etc.	Präsenzstudium • Lehrgespräch • Praktikum • Gruppendiskussionen • Präsentationen, etc.	
Unterrichtssprache	Deutsch			
Leistungsbewertung	Modulprüfung (70%), Kurztests (30%)			
Lehrmittel	**Barnes**, David, J.; Kölling, Michael: Java lernen mit BlueJ. Eine Einführung in die objektorientierte Programmierung, Pearson Studium, München, 2013, 5. Auflage, ISBN 978-3-86894-907-0 **Handouts** auf Moodle			
Vorkenntnisse: Modul(e)	-			
Anschlussmodul(e)	FTOOP, D&A, SWE			

*Studienniveau	**B** Basic level course: Modul zur Einführung in das Basiswissen eines Gebiets. **I** Intermediate level course: Modul zur Vertiefung der Basiskenntnisse. **A** Advanced level course: Modul zur Förderung und Verstärkung der Fachkompetenz. **S** Specialised level course: Modul zum Aufbau von Kenntnissen und Erfahrungen in einem Spezialgebiet.
**Typus	**C** Core course: Modul des Kerngebiets eines Studienprogramms. **R** Related course: Unterstützungsmodul zum Kerngebiet (z.B. Vermittlung von Vor- oder Zusatzkenntnissen). **M** Minor course: Wahl- oder Ergänzungsmodul.

1 Stoffplan

Das Schwergewicht des Moduls liegt in den grundlegenden Konzepten und Vorgehensweisen der objektorientierten Software-Entwicklung und weniger auf den spezifischen Einzelheiten einer Programmiersprache.

Neue Themen bzw. die entsprechenden Kapitel sind mit konkreten Projektbeispielen eng verknüpft.

Die Stoffvermittlung erfolgt problemorientiert: Zu Beginn wird ein Ziel festgelegt und eine Aufgabe formuliert. Im Verlauf der Analyse werden Konzepte, Sprachelemente und Konstrukte thematisiert, die zur Lösung erforderlich sind. Die neuen syntaktischen Themen werden also immer im Zusammenhang mit ihrer konkreten Anwendung eingeführt.

Objekte und Klassen

Konzepte: Objekte, Klassen, Methoden, Parameter

Klassendefinitionen

Konzepte: Datenfelder, Konstruktoren, Parameter, sondierende und verändernde Methoden, Zuweisung und bedingte Anweisung

Objektinteraktion

Konzepte: Abstraktion, Modularisierung, Objekterzeugung, Objektdiagramme, Methodenaufrufe, Debugger

Objektsammlungen

Konzepte: Sammlungen, Schleifen, Iteratoren, Arrays

Bibliotheksklassen nutzen

Konzepte: Benutzung von Bibliotheksklassen, Dokumentation lesen, Dokumentation schreiben

Fehlern vermeiden

Konzepte: Testen, Fehler beseitigen, Tests automatisieren

Klassenentwurf

Konzepte: Entwurf nach Zuständigkeiten, Kopplung, Kohäsion, Refactoring

Bessere Struktur durch Vererbung

Konzepte: Vererbung, Ersetzbarkeit, Subtyping, polymorphe Variablen

Mehr über Vererbung

Konzepte: Methoden-Polymorphie, statischer und dynamischer Typ, Überschreiben von Methoden, dynamische Methodensuche

Weitere Techniken zur Abstraktion

Konzepte: Abstrakte Klassen, Interfaces

Anonyme innere Klassen

Fehlerbehandlung

Konzepte: Defensive Programmierung, Exceptions auslösen und behandeln, Fehler melden, einfache Dateiverarbeitung