

Modulbezeichnung	<b>Praktische Informatik 2</b>								
Modulverantwortliche(r)	Dr. K. Hölscher								
Modulart	Pflicht/Wahl <input checked="" type="checkbox"/> Wahlpflicht <input type="checkbox"/>								
Spezialisierungsbereich									
Dauer des Moduls	1 Semester								
Kreditpunkte	9 CP								
Arbeitsaufwand	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%;">Berechnung des Workloads</td> <td></td> </tr> <tr> <td>Präsenz</td> <td style="text-align: right;">84 h</td> </tr> <tr> <td>Übungsbetrieb/Prüfungsvorbereitung</td> <td style="text-align: right;">186 h</td> </tr> <tr> <td>Summe</td> <td style="text-align: right; border-top: 1px solid black;">270 h</td> </tr> </table>	Berechnung des Workloads		Präsenz	84 h	Übungsbetrieb/Prüfungsvorbereitung	186 h	Summe	270 h
Berechnung des Workloads									
Präsenz	84 h								
Übungsbetrieb/Prüfungsvorbereitung	186 h								
Summe	270 h								
Turnus des Moduls	angeboten in jedem SoSe								
Voraussetzung für die Teilnahme	Keine <input checked="" type="checkbox"/> Folgende <input type="checkbox"/> Inhaltliche Voraussetzungen: Grundlagen der Programmierung								
Lehr- und Lernformen	Seminar <input type="checkbox"/> Vorlesung <input checked="" type="checkbox"/> Tutorium <input checked="" type="checkbox"/> Praktikum <input type="checkbox"/> Projekt <input type="checkbox"/>								
Lernziele	<ul style="list-style-type: none"> <li>• Grundlegende Konzepte der objektorientierten Programmierung kennen, verstehen und anwenden können.</li> <li>• Anschauliche Sachverhalte im Modell der Objektorientierung ausdrücken können.</li> <li>• Eine einfache Entwicklungsumgebung nutzen können.</li> <li>• LaTeX zur Erstellung einfacher Dokumente nutzen können.</li> <li>• Versionsverwaltungssysteme verstehen und einsetzen können.</li> <li>• Datenstrukturen und Algorithmen in Java umsetzen können</li> <li>• Fehler unter Einsatz eines einfachen Debuggers finden können.</li> <li>• Einfache Komponententests zur Qualitätssicherung erstellen und durchführen können.</li> <li>• Ein Softwaredokumentationswerkzeug verwenden können.</li> <li>• Typische Datenstrukturen identifizieren und problemadäquat einsetzen können.</li> <li>• Wesentliche Algorithmen der Informatik erklären, anwenden und modifizieren können.</li> <li>• Algorithmische Alternativen bezüglich der Eignung für ein Problem beurteilen können.</li> <li>• Die Komplexität von einfachen Algorithmen analysieren können.</li> <li>• In Gruppen Probleme analysieren und gemeinsam Lösungsstrategien entwickeln und präsentieren können.</li> </ul>								

Lerninhalte	<ol style="list-style-type: none"> <li>1. Prinzipien der objektorientierten Programmierung: Geheimnisprinzip – Methoden – Operationen – Objekte – Klassen – Botschaften – Ereignisverarbeitung – Attribute – Vererbung – Polymorphismus – Überladung – Generische Datentypen – Interfaces</li> <li>2. Datenstrukturen: Information und ihre Repräsentation – Datentypen und Typanalyse – Elementare und zusammengesetzte Datentypen – rekursive Datentypen</li> <li>3. Fehlervermeidung: Exceptions</li> <li>4. Dokumentation von Klassen, Methoden und Attributen</li> <li>5. Automatisierte Komponententests</li> <li>6. Fehlersuche (Debugging): Breakpoint – schrittweise Ausführung – Stacktrace</li> <li>7. Umsetzung der Punkte 1.-6. mit Java, Javadoc und JUnit</li> <li>8. Algorithmen: Begriff des Algorithmus – Beschreibung von Algorithmen – Algorithmische Umsetzung kanonischer Operationen auf Datenstrukturen – Grundlegende Strategien: Greedy, Divide-and-Conquer, Backtracking, dynamische Programmierung, zufallsgesteuerte Algorithmen, genetische Algorithmen, heuristische Algorithmen, probabilistische Algorithmen</li> <li>9. Komplexität von Algorithmen – <math>O(n)</math>-Notation und asymptotische Analyse</li> <li>10. Suchen und Sortieren auf Arrays: Binäre Suche – Quicksort und weitere Sortieralgorithmen – Komplexitätsvergleiche</li> <li>11. Listen – Stapel – Warteschlangen: Datenstrukturen zur Realisierung (Arrays versus Verkettung und dynamische Speicherallokation für Elemente), Algorithmen zur Realisierung kanonischer Operationen (Listentraversal, Anfügen, Einfügen, Löschen, Suchen, Stack-Operationen, FIFO-Warteschlangenoperationen)</li> <li>12. Bäume: Binäre Bäume, AVL-Bäume, Rot-Schwarz-Bäume, B-Bäume – Suchen, Einfügen, Löschen, Traversal</li> <li>13. Hashing: Hash-Array, Hashfunktion, Hash Buckets, offenes Hashing</li> <li>14. Graphen: ungerichtete, gerichtete, gewichtete Graphen – Repräsentation durch Knoten- und Kantenlisten, durch Adjazenzmatrizen, Adjazenzlisten – Algorithmen auf Graphen: Breitensuche, Tiefensuche, kürzeste Wege auf gewichteten Graphen: Dijkstras Algorithmus, minimal aufspannende Bäume: Algorithmen von Prim et al. und Kruskal</li> </ol> <p>Im Rahmen des Übungsbetriebes werden <math>\LaTeX</math> und Versionskontrolle mittels Git eingeführt und verwendet.</p> <p>Lehrveranstaltung(en):</p> <ul style="list-style-type: none"> <li>● 03-B-MI-22.1 Objektorientierte Programmierung [OOP] (3 CP)</li> <li>● 03-B-MI-22.2 Algorithmen und Datenstrukturen [AuD] (6 CP)</li> </ul>
Prüfungsformen	KP; PL1: 30%, PL2: 55%, PL3: 15% ; Klausur, Portfolio, Fachgespräch
Literatur	<ul style="list-style-type: none"> <li>● David J. Barnes, Michael Kölling: Java lernen mit BlueJ - Objects first - Eine Einführung in Java. Aktuelle Auflage. Pearson Studium.</li> <li>● Christian Ullenboom: Java ist auch eine Insel. Aktuelle Auflage. Rheinwerk Computing.</li> <li>● Thomas Ottmann, Peter Widmayer: Algorithmen und Datenstrukturen. Aktuelle Auflage, Spektrum Akademischer Verlag.</li> <li>● Robert Sedgewick, Robert Wayne: Algorithmen. Aktuelle Auflage. Pearson Studium.</li> <li>● Markus von Rimscha: Algorithmen kompakt und verständlich. Aktuelle Auflage. Springer Vieweg.</li> </ul>