Einleitung

Studierende, die das Modul „Einführung in die Programmierung“ (PROG) belegt haben, darüber hinaus aber keinerlei Erfahrung mit Programmierung haben und auch in ihrem beruflichen Alltag damit nicht in Berührung kommen, tun sich oft sehr schwer mit dem Modul „Fortgeschrittene Programmiertechniken“ (FOPT). Aus diesem Grund wurde dieser Vorkurs eingerichtet, der Ihnen den Übergang von PROG zu FOPT erleichtern soll.
Ich habe vier Themenbereiche herausgesucht, deren Behandlung mir am dringlichsten erscheint:

- Ablauf eines Java-Programms
- Rekursion
- Generics
- Schnittstellen, Lambda-Ausdrücke und Methodenreferenzen

Sie sollen durch diesen Vorkurs folgende Lernziele erreichen:

- Sie sollen verstehen, was beim Ablauf eines Java-Programms passiert. Dies umfasst vor allem ein Verständnis für drei wichtige Speicherbereiche, die für die Ausführung eines Java-Programms genutzt werden:
    - Sie sollen erläutern können, dass auf dem Stack pro Methodenaufruf ein Methodenaktivierungsblock angelegt wird, der die übergebenen Parameter, die Rücksprungadresse, einen Speicherbereich für den Rückgabewert sowie für die lokalen Variablen enthält.
    - Sie sollen erklären können, dass für neu erzeugte Objekte Speicherplatz im Heap belegt wird, dass dieser Speicherbereich Platz zum Speichern der Attributwerte des Objekts bereitstellt, und dass lokale Variable und Attribute, deren Typ kein primitiver Datentyp ist, immer Referenzen auf die entsprechenden Objekte enthalten. Dass sich auch der Speicherplatz für Felder im Heap befindet, sollte Ihnen geläufig sein. Insbesondere sollen Sie verstehen, welche Struktur mehrdimensionale Felder im Heap besitzen.
    - Sie sollen schließlich darstellen können, dass es für statische Attribute einen eigenen statischen Speicherbereich gibt.

  Sie sollen selbstständig basierend auf einem Java-Programm eine Skizze anfertigen können, die den Zustand des Stacks, des Heaps und des statischen Speicherbereichs zu einem vorgegebenen Zeitpunkt beim Ablauf des Programms darstellt. Dabei sollen Sie auch den Unterschied zwischen statischen und nicht-statischen Methoden beim Aufruf erklären können. Schließlich werden Sie durch dieses Hintergrundwissen besser und gezielter mit einem Debugger arbeiten, den Unterschied zwischen Call-by-value und Call-by-reference anhand von Beispielen erläutern sowie das Prinzip der Abfallsammlung (Garbage Collection) darstellen können.

- Sie sollen über zahlreiche Beispiele die Scheu vor der Rekursion verlieren. Insbesondere sollen Sie durch die Kenntnisse aus dem vorigen Kapitel erläutern können, was bei einem rekursiven Methodenaufruf vor sich geht. Sie sollen die Bedeutung eines Rekursionsankers für das Ende einer Rekursion darstellen und unterschiedliche Varianten von Rekursion aufzählen können wie z.B. direkte und indirekte Rekursion. Sie sollen eigene Programme mit rekursiven Methoden erstellen können, vorwiegend zum Durchlaufen von Bäumen, zum Backtracking oder zur Realisierung des Prinzips „Teile und herrsche“. Sie sollen dazu erklären können, auch anhand konkreter Beispiele, dass es beim Backtracking und „Teile und herrsche“ auch um das Durchlaufen von Bäumen geht, die allerdings als Datenstruktur nicht explizit existieren müssen.
- Sie sollen verstehen, dass das Generics-Konzept eine Parametrisierung „auf höherer Ebene“ darstellt, indem in einer Klasse oder Schnittstelle ein oder mehrere Typen nicht festgelegt, sondern noch offen gehalten werden, und erst später, etwa bei der Erzeugung eines Objekts dieser Klasse oder bei der Implementierung einer Schnittstelle durch eine Klasse, konkretisiert werden können. Sie sollen eigene generische Klassen und Schnittstellen programmieren können und dabei die generischen Typen auch sinnvoll einschränken können. Sie sollen anhand eines kleinen Einblicks in die Umsetzung des Generics-Konzepts den Begriff „Type Erasure“ erklären können. Sie sollen verstehen, dass die allgemein übliche Typkompatibilität, die für Typen außerhalb der spitzen Klammern (< und >) gilt, nicht für die Typen innerhalb der spitzen Klammern gilt, und dass es zu diesem Zweck sogenannte Wildcards gibt, deren Typ durch extends „nach oben“ und durch super „nach unten“ begrenzt werden kann. Sie sollen damit entscheiden können, welche Zuweisungen bzw. Parameterübergaben im Zusammenhang mit Generics möglich sind und welche nicht. Schließlich sollen Sie auch noch wissen, dass nicht nur Klassen und Schnittstellen generisch sein können, sondern auch Methoden. Sie sollen dadurch in der Lage sein, eigene generische Methoden zu schreiben.
- Sie sollen verstehen, dass Schnittstellen nicht nur von Klassen implementiert werden, sondern sinnvollerweise auch als Typen von Variablen, Methodenparametern usw. vorkommen können. Dies sollen Sie insbesondere anhand der Beispielprogramme Proxy, Template und Command erläutern und dabei diese Programme variieren können. Sie sollen Unterschiede zwischen Schnittstellen und abstrakten Klassen mit Hilfe von Beispielen erklären können. Sie sollen verstehen, dass Schnittstellen nicht nur abstrakte Methoden, sondern auch Methoden mit einer Implementierung enthalten können, und zwar in Form von statischen Methoden oder Default-Methoden. Sie sollen wissen, was funktionale Schnittstellen sind

und wann Lambda-Ausdrücke verwendet werden können. Sie sollen die syntaktischen Varianten von Lambda-Ausdrücken kennen und in der Lage sein, Lambda-Ausdrücke sowohl zu verstehen als auch selbst schreiben zu können. Sie sollen wissen, dass unter besonderen Umständen Lambda-Ausdrücke durch Methodenreferenzen ersetzt werden können. Sie sollen Lambda-Ausdrücke in Methodenreferenzen umwandeln können, sofern dies möglich ist, sowie umgekehrt aus Methodenreferenzen die dazu äquivalenten Lambda-Ausdrücke machen können.

Der Lehrtext enthält neben den vier Hauptkapiteln mit den soeben dargestellten Lernzielen noch einige Informationen über weitere Themen. Es geht dabei um Collections (Listen, Mengen und Maps), um die Ausnahmebehandlung, um Records und Enums sowie um Packages und Module. Alle Themen werden nur kurz angerissen und stellen lediglich eine kleine Einführung in die jeweilige Thematik dar. Der Text soll Sie dazu anregen, sich über andere Quellen weitere Informationen zu beschaffen, falls Sie bei sich Wissensdefizite in einem oder mehreren dieser Bereiche entdecken.