Vorwort

Dieses Buch gibt eine Einführung in die funktionale Programmierung in der Sprache Java. Wir kennen Java als eine Sprache, die auf dem objektorientierten Programmierparadigma beruht und wir sind gut mit dieser Art der Programmierung vertraut. Und wir wissen, dass die in Java 8 eingeführten Lambda-Ausdrücke die Grundlage für eine funktionale Programmierung bilden. Aber rechtfertigt dieses neue Sprachfeature ein Buch mit 300 Seiten?

Wie wir sehen werden, unterscheidet sich funktionale Programmierung grundsätzlich von unserer gewohnten Welt der imperativen und objektorientierten Programmierung. Schon die Ursprünge sind gänzlich unterschiedlich. Während imperative Programmierung als Abstraktion von Maschinencode und objektorientierte Programmierung aus der Motivation, Dinge der Realität abzubilden, entstanden sind, wurde funktionale Programmierung auf Basis einer mathematischen Theorie, dem Lambda-Kalkül, geschaffen. Mehr erfahren Sie dazu in Kapitel 1 dieses Buches.

Für eine funktionale Programmierung muss man sich daher auf andere Denkweisen, Prinzipien und Programmiermuster einlassen. Dieses Buch will diese Denkweisen, Prinzipien und Programmiermuster mit ihrer Ausgestaltung in Java und die darauf aufbauenden Techniken und Systeme vermitteln. Es orientiert sich dabei stark an den vielfältigen Ideen, Konzepten und Anwendungen, die die Forschung zur funktionalen Programmierung in 60 Jahren hervorgebracht hat. In dem Sinne kann das Buch auch als ein Versuch gesehen werden, dieses vielfältige Wissen auf eine Programmierung in Java zu übertragen. Das Buch verfolgt daher sehr explizit das Ziel, nicht nur die Sprachkonzepte und die neuen Systeme und Bibliotheken zu beschreiben, sondern ganz besonders auch die grundlegenden Ideen und Prinzipien zu vermitteln und damit ein tieferes Verständnis für die funktionale Programmierung zu schaffen.

Das Buch hat seinen Ursprung in Lehrveranstaltungen zur funktionalen Programmierung, die ich seit mehreren Jahren an der Johannes Kepler Universität Linz halte. Dazu gehört eine Speziallehrveranstaltung zur funktionalen Programmierung in Java und eine Lehrveranstaltung zu den Prinzipien der funktionalen

Programmierung, die sich auf die Konzepte der rein-funktionalen Sprache Haskell stützt. Eine weitere wichtige Erfahrung, die in dieses Buch eingeflossen ist, war die Beschäftigung und das Arbeiten mit der Programmiersprache Scala. Scala sehe ich als Vorbild für einen Sprachentwurf, der objektorientierte und funktionale Konzepte systematisch vereint. In diesem Sinne war Scala in vielerlei Hinsicht auch Vorbild für die Gestaltung der funktionalen Konzepte in Java.

Zielgruppe

Das Buch richtet sich an Softwareentwickler bzw. Studenten der Informatik, die bereits Erfahrung mit objektorientierter Programmierung in Java haben und sich in das neue Paradigma der funktionalen Programmierung einarbeiten wollen. Die umfassende Behandlung des Themas sollte den Erwerb von fundierten Kenntnissen der funktionalen Programmierung in Java ermöglichen. Das Buch kann aber auch verwendet werden, um sich gezielt Kenntnisse von spezifischen Techniken anzueignen. Der folgende Abschnitt »Pfade durch das Buch« informiert über mögliche Kapitelfolgen beim Lesen.

Das Buch eignet sich auch besonders als begleitendes Lehrbuch zu Vorlesungen zum Thema funktionale Programmierung in Java. Des Weiteren würde ich das Buch als ergänzendes Lehrbuch zu einer Lehrveranstaltung zu fortgeschrittenen Programmiertechniken in Java oder zu einer sprachunabhängigen Einführung in die funktionale Programmierung empfehlen.

Aufbau des Buchs

Das Buch teilt sich konzeptionell in drei Teile: Im ersten Teil, der die Kapitel 1 und 2 umfasst, werden die Grundlagen zu einer funktionalen Programmierung gelegt. Der zweite Teil umfasst die Kapitel 3 bis 6 und vermittelt wichtige Prinzipien der funktionalen Programmierung in Java. Im dritten Teil mit den Kapiteln 7 bis 11 werden wichtige Techniken, Systeme und Bibliothekskomponenten beschrieben. Das Buch schließt in Kapitel 12 mit einem Blick auf die derzeitigen Grenzen der funktionalen Programmierung in Java und mögliche zukünftige Entwicklungen.

Im Einzelnen behandeln die Kapitel folgende Themen:

In Kapitel 1 wird eine allgemeine Einführung in die funktionale Programmierung inklusive der geschichtlichen Entwicklung gegeben.

Kapitel 2 behandelt die sprachlichen Grundlagen von Java für die funktionale Programmierung. Dazu gehören nicht nur die Lambda-Ausdrücke, sondern auch Generics sowie die neuen Default-Methoden bei Interfaces.

Kapitel 3 befasst sich mit der Programmierung ohne Seiteneffekte. In diesem Kapitel werden auch elementare funktionale Datenstrukturen eingeführt, die in Folge immer wieder verwendet werden.

Kapitel 4 geht umfassend auf das Arbeiten mit Funktionsparametern ein. Anhand einer Reihe von Fallbeispielen wird gezeigt, wie mit Funktionsparametern Methoden mit hohem Abstraktionsgrad realisiert werden können.
Kapitel 5 befasst sich mit der Kombination von Funktionen. Es werden die Prinzipien und Möglichkeiten dargestellt, wie man mit Kombinationsoperatoren aus einfachen funktionalen Bausteinen komplexe Funktionen aufbauen kann.
Im Kapitel 6 werden die in den vorherigen Kapiteln gefundenen Prinzipien und Programmmuster in einen allgemeinen Kontext gestellt. Es wird sich zeigen, dass sich dahinter sehr allgemeine, breit einsetzbare Strukturen verbergen, die ihren Ursprung in der Mathematik haben und damit entsprechende formale Eigenschaften mitbringen.
Kapitel 7 ist das erste Kapitel zu Techniken und Systemen und enthält eine detaillierte Einführung in das Arbeiten mit den funktionalen Streams.
In einem eigenen Kapitel 8 gehen wir dann auf die parallele Ausführung bei Streams ein.
In Kapitel 9 wird die Implementierung asynchroner Tasks auf Basis von CompletableFutures behandelt.
Kapitel 10 gibt eine detaillierte Einführung in das System RxJava, das eine Implementierung der reaktiven Streams für eine Programmierung von asynchronen Ereignisströmen bereitstellt.
Kapitel 11 enthält eine kurze Darstellung der Anwendung funktionaler Methoden bei Unit-Tests sowie eine Einführung in eigenschaftsbasiertes Testen nach der Methode von QuickCheck.
Im abschließenden Kapitel 12 wird ein Einblick auf weiterführende Konzepte der funktionalen Programmierung gegeben, wie sie in den Sprachen Scala und Kotlin zur Verfügung stehen. Es gibt auch Hinweise auf Features, die bei Java für zukünftige Releases geplant sind.
Der Appendix beinhaltet die Daten zu den Experimenten, die die Einflüsse unterschiedlicher Faktoren auf die Laufzeit von parallelen Streams zeigen.

Pfade durch das Buch

Abhängig von Interesse und Vorkenntnissen bieten sich mehrere Varianten für die Abfolge der Kapitel dieses Buches an. Auch ein Lesen einzelner Themen ist möglich. In den einzelnen Kapiteln sind des Öfteren Verweise zu Inhalten und Beispielen aus früheren Kapiteln gegeben, die man dann eventuell zielgerichtet nachlesen kann.

Des Weiteren beinhaltet Kapitel 5 zwei Abschnitte mit fortgeschrittenen Anwendungsbeispielen. Diese Abschnitte können bei der Erstlektüre übersprungen werden, ohne dass das Lesen der folgenden Kapitel beeinträchtigt wird. Auf

diese Möglichkeit wird bei den Abschnitten in der Form von Fußnoten hingewiesen.

Im Folgenden sind mehrere mögliche Kapitelfolgen angegeben.

Vollständiger Pfad

Es ist offenkundig, dass ein Lesen der Kapitel 1 bis 12 empfohlen wird, wobei eventuell die gekennzeichneten Abschnitte mit fortgeschrittenen Themen ausgespart werden können. Mit dem Lesen der Kapitel 3 bis 6, in denen die allgemeinen Prinzipien der funktionalen Programmierung vermittelt werden, soll auch ein Verständnis für die Gestaltung der in den Kapiteln 7 bis 11 dargestellten Techniken und Systeme geschaffen werden.

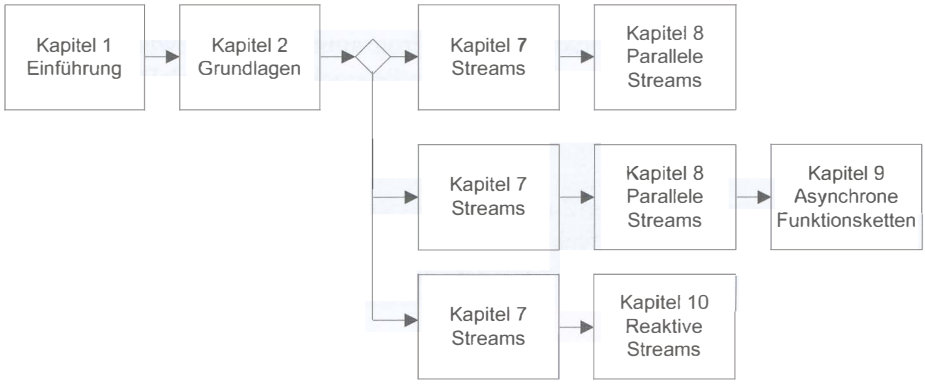
Techniken

Ein weiterer Pfad bietet sich an, wenn man primär an den bekannten Techniken und Systemen und weniger an den allgemeinen Prinzipien der funktionalen Programmierung in Java interessiert ist. Hier besteht die Möglichkeit, mit Kapitel 1 und 2 die Grundlagen zu erwerben und dann mit den Techniken und Systemen in den Kapiteln 7 bis 11 fortzufahren. Bei Verweisen zu Inhalten aus früheren Kapiteln kann man diese zielgerichtet nachlesen.

Einzelne Techniken

Das Buch bietet auch die Möglichkeit, sich einzelne Techniken zu erarbeiten. Man startet wieder mit Kapitel 1 und 2 zu den Grundlagen und kann dann:

für Streams mit dem Kapitel 7 gefolgt von Kapitel 8 fortsetzen
für asynchrone Ausführungsmodelle das Kapitel 9 lesen, wobei hier die parallelen Streams vorausgesetzt werden und daher das Lesen von Kapitel 7 und 8 empfohlen wird
für die reaktiven Streams direkt zum Kapitel 10 gehen, wobei hier entweder Vorkenntnisse zu den Streams oder das Lesen von Kapitel 7 angeraten ist.



Konventionen

In diesem Buch gelten folgende Konventionen bezüglich Schriftarten und Schreibweisen.

Programme und Programmelemente in Fixpunkt-Font

Für alle Programmbeispiele wird ein Fixpunkt-Font verwendet. Ebenso werden Bezeichner von Programmelementen innerhalb des Fließtextes in Fixpunkt-Font gesetzt. Wird aber der Begriff allgemein verwendet, wird er mit normaler Schriftart geschrieben. Zum Beispiel wird der Begriff String für Zeichenketten mit normaler Schriftart geschrieben, wird aber die Klasse `String` explizit genannt, wird für den Klassennamen der Fixpunkt-Font verwendet.

Hervorhebungen

Besondere Begriffe werden bei der Einführung *kursiv* geschrieben.

Programmbeispiele

Größere Programmbeispiele, besonders Code von Klassen, werden mit Unterschrift versehen und mit der Kategorie *Listing* nummeriert. Kürzere Anweisungsfolgen werden ohne Nummerierung direkt in den Fließtext eingefügt.

Fachbegriffe und englische Bezeichner

Wenn immer möglich und sinnvoll, werden im Buch deutsche Begriffe verwendet. Bei manchen Begriffen wird aber auf englische Bezeichner zurückgegriffen. So wird zum Beispiel *Map* und nicht *Abbildung* für Programmkomponenten verwendet, die eine Abbildung von Schlüssel auf Werte realisieren. Englische Bezeichner werden dann großgeschrieben, zum Beispiel Streams. Zusammensetzun-

gen aus englischen und deutschen Wörtern werden mit Bindestrich verbunden, zum Beispiel Fixpunkt-Font. Elemente von Programmen werden wie in Java mit Groß- und Kleinschreibung geschrieben, zum Beispiel CompletableFuture.

Anmerkungen und Hinweise

Das Buch gibt an einigen Stellen Zusatzinformationen, Anmerkungen und Hinweise, die über das unmittelbare Thema funktionale Programmierung in Java hinausgehen. Diese sind durch graue Blöcke besonders gekennzeichnet.

Sourcecode zu den Programmbeispielen

Der Sourcecode zu allen Programmbeispielen kann von der Webseite zu diesem Buch

`www.dpunkt.de/fpinjava`

heruntergeladen werden. Es finden sich hier mehrere Maven-Projekte. Der Code ist in Packages analog zu den Abschnitten des Buchs organisiert.

Danksagung

Zu diesem Buch haben eine Reihe von Personen beigetragen, bei denen ich mich hier ausdrücklich bedanken möchte.

Ich möchte zuerst meiner Lektorin beim dpunkt.verlag, Frau Melanie Feldmann, für die gründliche und kompetente Lektoratsarbeit danken. Auch möchte ich den anonymen Gutachtern für ihr wertvolles Feedback danken.

Mein besonderer Dank gilt dem Leiter des Instituts für Systemsoftware, Professor Hanspeter Mössenböck, der mir an seinem Institut die Möglichkeit und das Umfeld geboten hat, mich so eingehend mit dem Thema funktionale Programmierung zu beschäftigen. Ich danke ihm auch für das Herstellen des Kontakts zum dpunkt.verlag. Mein Dank geht auch an meine Kollegen am Institut für die immer angenehme und fördernde Arbeitsatmosphäre. Meine Kollegen Kevin Feichtinger und Daniel Hinterreiter waren sehr hilfreich, indem sie erste Versionen von Kapiteln dieses Buches lasen und mir wichtige Rückmeldungen gaben. Meinem Kollegen Professor Paul Grünbacher danke ich für die vielen Jahre der freundschaftlichen Zusammenarbeit und für sein wertvolles Feedback zur Verbesserung der Einleitung zu diesem Buch. Bedanken möchte ich mich auch bei den Kollegen aus dem Oracle-Team, Dr. Roland Schatz und Dr. Lukas Stadler, die mir zu allen Details zur Sprache Java und ihrer Ausführung immer äußert kompetent helfen konnten.

Meinem ehemaligen Kollegen und Freund Dr. Alexander Fried danke ich ganz besonders für seinen Beitrag zur Gestaltung an und seine Mitarbeit bei unserer

gemeinsamen Lehrveranstaltung »Functional Programming in Java«. Ich danke ihm auch, dass er mich immer ermutigt hat, das Buchprojekt in Angriff zu nehmen.

Schließlich möchte ich mich bei einer Reihe von Studenten bedanken, die durch ihre hervorragenden Studienarbeiten wertvollen Input zu diesem Buch geliefert haben. Namentlich möchte ich hier nennen (in alphabetischer Reihenfolge): Christoph Burghuber, Christoph Gerstberger, Marcel Homolka, Florian Latifi, Daniel Schneider und Florian Schrögendorfer.

Herbert Prähofer
Linz, April 2020