

1 Einleitung

Bevor es mit den Programmierthemen richtig losgeht, möchte ich Ihnen dieses Buch kurz vorstellen. Dabei beginne ich damit, warum dieses Buch entstanden ist und wie es Ihnen hoffentlich helfen kann, ein noch besserer Java-Entwickler zu werden. Danach folgt eine etwas ausführlichere Vorstellung der Gliederung des Inhalts, damit Sie sich sofort im Buch zurechtfinden.

1.1 Über dieses Buch

1.1.1 Motivation

Mein Ziel war es, ein Buch zu schreiben, wie ich es mir selbst immer als Hilfe gewünscht habe. Die hier vorgestellten Hinweise und Techniken sollen Sie auf Ihrem Weg vom engagierten Hobbyprogrammierer oder Berufseinsteiger zum erfahrenen Softwareentwickler begleiten. Dieser Weg ist ohne Anleitung gewöhnlich steinig und mit einige Mühen, Irrwegen und Problemen verbunden. Einige dieser leidvollen Erfahrungen möchte ich Ihnen ersparen. Aber auch erfahreneren Softwareentwicklern soll dieses Buch eine Möglichkeit geben, über die im täglichen Einsatz lieb gewonnenen Gewohnheiten nachzudenken und die eine oder andere davon zu ändern, um die Produktivität weiter zu steigern. Mein Wunsch ist, dass sich nach Lektüre des Buchs für Sie die Ingenieurdisziplin der Softwareentwicklung mit der Kunst des Programmierens verbindet und Dinge auf einmal einfach so auf Anhieb funktionieren. Das ist etwas ganz anderes, als vor jedem Gang in die Testabteilung Magenschmerzen zu bekommen.

Sowohl der Berufseinstieg als auch die tägliche Arbeit können manchmal frustrierend sein. Meiner Meinung nach soll Softwareentwicklung aber Spaß und Freude bereiten, denn nur so können wir exzellente Resultate erzielen. In der Praxis besteht jedoch die Gefahr, in die Fettnäpfchen zu treten, die im Sourcecode hinterlassen wurden. Dies geschieht meistens dadurch, dass die existierende Lösung softwaretechnisch umständlich oder schlecht implementiert ist und/oder nicht bis zu Ende durchdacht wurde. Der Sourcecode ist dann häufig schwierig wart- und erweiterbar. Manchmal bereitet bereits das Auffinden der Stelle, an der man Modifikationen durchführen sollte, Probleme.

Dieses Buch soll aufzeigen, wie man die zuvor beschriebene »Altlasten«-Falle vermeidet oder aus ihr herauskommt und endlich Sourcecode schreiben kann und darf, der

leicht zu lesen ist und in dem es Spaß macht, Erweiterungen zu realisieren. Grundlage dafür ist, dass wir uns einen Grundstock an Verhaltensweisen und an Wissen aneignen.

1.1.2 Was leistet dieses Buch und was nicht?

Wieso noch ein Buch über Java-Programmierung? Tatsächlich kann man sich diese Frage stellen, wo es doch unzählige Bücher zu diesem Thema gibt. Bei vielen davon handelt es sich um einführende Bücher, die häufig lediglich kurz die APIs anhand simpler Beispiele vorstellen. Die andere große Masse der Java-Literatur beschäftigt sich mit speziellen Themen, die für den »erfahrenen Einsteiger« bereits zu komplex geschrieben und in denen zu wenig erklärt ist. Genau hier setzt dieses Buch an und wagt den Spagat, den Leser nach der Lektüre einführender Bücher abzuholen und so weit zu begleiten, dass er mit einem guten Verständnis die Spezialliteratur lesen und gewinnbringend einsetzen kann.

Ziel dieses Buchs ist es, dem Leser fundierte Kenntnisse in Java und einigen praxisrelevanten Themenbereichen, unter anderem dem Collections-Framework und im Bereich Multithreading, zu vermitteln. Es werden vertiefende Blicke auf die zugrunde liegenden Details geworfen, um nach Lektüre des Buchs professionelle Programme schreiben zu können. Wie bereits angedeutet, bietet dieses Buch keinen Einstieg in die Sprache selbst, sondern es wird bereits einiges an Wissen oder Praxiserfahrung vorausgesetzt. In zwei einleitenden Grundlagenkapiteln über objektorientiertes Design und Java wird allerdings die Basis für das Verständnis der Folgekapitel geschaffen.

In diesem Buch versuche ich, einen lockeren Schreibstil zu verwenden und nur an den Stellen formal zu werden, wo dies wirklich wichtig ist, etwa bei der Einhaltung von Methodenkontrakten. Da der Fokus dieses Buchs auf dem praktischen Nutzen und dem guten Verständnis von Konzepten liegt, werden neben APIs auch häufig vereinfachte Beispiele aus der realen Welt vorgestellt. Die meisten der abgebildeten Listings stehen als kompilierbare und lauffähige Programme (Ant-Targets) auf der Webseite zum Buch zum Download bereit. Im Buch selbst werden aus Platzgründen und zugunsten einer besseren Übersichtlichkeit in der Regel nur die wichtigen Passagen abgedruckt.

1.1.3 Wie und was soll mithilfe des Buchs gelernt werden?

Dieses Buch zeigt und erklärt einige in der Praxis bewährte Ansätze, Vorgehens- und Verhaltensweisen, ohne dabei alle Themengebiete bis in kleinste Detail auszuleuchten. Wichtige Details bzgl. Algorithmen und Datenstrukturen werden jedoch bei Bedarf eingeführt. Es wird der pragmatische Weg gegangen und bevorzugt die in der täglichen Praxis relevanten Themen vorgestellt. Sollte ein Thema bei Ihnen besonderes Interesse wecken und sie weitere Informationen wünschen, so finden sich in den meisten Kapiteln Hinweise auf weiterführende Literatur. Dies ist im Prinzip auch schon der erste Tipp: ***Lies viele Bücher und schaffe damit eine breite Wissensbasis.*** Ich zitiere hier aus Jon Bentley's Buch »Perlen der Programmierkunst« [5]: ***»Im Stadium des Entwurfsprozesses ist es unschätzbar, die einschlägige Literatur zu kennen.«***

Diesem Hinweis kann ich mich nur anschließen und möchte Ihnen hier speziell einige – meiner Meinung nach – ganz besondere Bücher ans Herz legen und empfehle ausdrücklich, diese Bücher begleitend oder ergänzend zu diesem Buch zu lesen:

- »**SCJP – Sun Certified Programmer & Developer for Java 2**« [63] – Die Vorbereitung zur Zertifizierung als Java-Programmierer mit all seinen Fallstricken und kniffligen Details wird auf unterhaltsame Weise von Kathy Sierra und Bert Bates aufbereitet.
- »**The Java Programming Language**« [2] – Ein unglaublich gutes Buch von Ken Arnold, James Gosling und David Holmes über die Sprache Java, das detailreich, präzise und dabei angenehm verständlich zu lesen ist.
- »**Effective Java**« [7] und [8] – Dieses Buch von Joshua Bloch habe ich auf der Java One 2001 in San Francisco gekauft und es hat mein Denken und Programmieren in Java stark beeinflusst. Mittlerweile existiert eine zweite Auflage, die auf JDK 6 aktualisiert wurde.
- »**Entwurfsmuster**« [25] – Das Standardwerk der sogenannten »Gang of Four« (Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides) habe ich 1998 kennengelernt und mit großem Interesse gelesen und gewinnbringend eingesetzt. Die vorgestellten Ideen sind beim Entwurf guter Software enorm hilfreich.
- »**Head First Design Patterns**« [23] – Dieses Buch einer anderen »Gang of Four« (Eric Freeman, Elizabeth Freeman, Kathy Sierra und Bert Bates) lässt Entwurfsmuster als ein unterhaltsames Thema erscheinen und ermöglicht einen guten Einstieg.
- »**Refactoring**« [22] – Einige Tricks und Kniffe zur Verbesserung von Sourcecode lernt man von Kollegen oder durch Erfahrung. Martin Fowler fasst dieses Wissen in dem genannten Buch zusammen und stellt ein systematisches Vorgehen zur Sourcecode-Transformation vor.
- »**Refactoring to Patterns**« [43] – Dieses Buch von Joshua Kerievsky verknüpft das Standardwerk zu Refactorings von Martin Fowler mit den Ideen der Entwurfsmuster.
- »**Code Craft: The Practice of Writing Excellent Code**« [27] – Ein lesenswertes Buch von Pete Goodlife, das diverse gute Hinweise gibt, wie man exzellenten Sourcecode schreiben kann, der zudem (nahezu) fehlerfrei, gut testbar sowie einfach zu warten ist.

Lesen hilft uns bereits, aber nur durch Übung und Einsatz in der Praxis können wir unsere Fähigkeiten verbessern. Weil ein Buch jedoch nicht interaktiv ist, werde ich bevorzugt eine schrittweise Vorstellung der jeweiligen Themen vornehmen, wobei zum Teil auch bewusst zunächst ein Irrweg gezeigt wird. Anhand der vorgestellten Korrekturen erkennt man dann die Vorteile viel deutlicher, als wenn nur eine reine Präsentation der Lösung erfolgen würde. Mit dieser Darstellungsweise hoffe ich, dass Sie sich ein paar gute Gewohnheiten antrainieren. Das fängt mit scheinbar einfachen Dingen wie der Vergabe von sinnvollen Namen für Variablen, Methoden und Klassen an und endet

in der Verwendung von problemangepassten Entwurfsmustern. Anfangs erfordert dies erfahrungsgemäß ein wenig Fleiß, Einarbeitung, Disziplin und eventuell sogar etwas Überwindung. Daher werde ich bei der Vorstellung einer Technik jeweils sowohl auf die Vorteile als auch die Nachteile (wenn vorhanden) eingehen.

1.1.4 Wer sollte dieses Buch lesen?

Dieses Buch konzentriert sich auf Java als Programmiersprache – allerdings benötigen Sie bereits einige Erfahrung mit Java, um die Beispiele sowie die beschriebenen Tücken nachvollziehen zu können und möglichst viel von den Tipps und Tricks in diesem Buch zu profitieren. Wenn Sie dieses Buch in den Händen halten, gehe ich davon aus, dass Sie sich bereits (etwas) mit Java auseinandergesetzt haben.

Das Buch richtet sich im Speziellen an zwei Zielgruppen: Zum einen sind dies engagierte Hobbyprogrammierer, Informatikstudenten oder Berufseinsteiger, die von Anfang an lernen wollen, wie man professionell Software schreibt. Zum anderen sind dies erfahrenere Softwareentwickler, die ihr Wissen in einigen fortgeschritteneren Themen komplettieren wollen und vermehrt Priorität auf sauberes Design legen oder Coding Conventions, Codereviews und Unit-Testen bei der Arbeit etablieren wollen.

Abhängig vom Kenntnisstand zu Beginn der Lektüre starten Entwickler mit Erfahrung bei Teil II oder Teil III des Buchs und können die dort vorgestellten Techniken sofort gewinnbringend in der Praxis einsetzen. Allen Lesern mit noch relativ wenig Erfahrung empfehle ich, den ersten Teil konzentriert und vollständig durchzuarbeiten, um sich eine gute Basis zu verschaffen. Dadurch wird das Verständnis der später vorgestellten Themen erleichtert, denn die nachfolgenden Teile des Buchs setzen die Kenntnis dieser Basis voraus und das Niveau nimmt ständig zu. Ziel ist es, nach Lektüre des Buchs, den Einstieg in die professionelle Softwareentwicklung mit Java erreicht zu haben und viele dazu erforderliche Techniken sicher zu beherrschen. Das Buch ist daher mit diversen Praxistipps gespickt, mit denen Sie auf interessante Hintergrundinformationen oder auf mögliche Probleme hingewiesen werden und die wie folgt in den Text integriert sind:

Tipp: Praxistipp

In derart formatierten Kästen finden sich im späteren Verlauf des Buchs immer wieder einige wissenswerte Tipps und ergänzende Hinweise zum eigentlichen Text.

1.2 Aufbau des Buchs

Der Aufbau des Buchs orientiert sich am Prozess der Softwareentwicklung, der aus verschiedenen Aktivitäten und Phasen besteht.

- **Analyse und Design** – In dieser Phase entsteht eine erste Beschreibung der wichtigsten Abläufe und der Struktur des zu erstellenden Systems basierend auf einer Anforderungs- und Analysephase mit Kunden. Es wird zudem festgelegt, »wie« die einzelnen Aufgaben realisiert werden können.
- **Implementierung** – Diese Phase besteht aus schrittweisem Implementieren des zuvor entworfenen Systems mit kleineren nachfolgenden oder begleitenden Tests. Bei größeren Systemen erfolgt zu bestimmten Zeitpunkten eine Integration in bereits bestehende Komponenten oder Systeme.
- **Qualitätssicherung und Test** – Während der Phase des Tests erfolgt eine Untersuchung der Software auf ein mögliches funktionales und anderweitiges Fehlverhalten, etwa nicht angemessene Antwortzeiten.

Einige der genannten Aktivitäten und Phasen müssen nicht unbedingt nacheinander durchgeführt werden, sondern können durchaus parallel erfolgen. Gerade aktuellere Vorgehensmodelle betonen das iterative Entstehen von Software und propagieren eine Überschneidung einzelner Phasen. Sinnvoll ist dies vor allem für die Implementierung und die Qualitätssicherung. Kleinere Tests können und sollten während der Entwicklung stattfinden. Details zu den einzelnen Phasen liefert Anhang B.

1.2.1 Gliederung des Buchs

Den Phasen der Softwareentwicklung folgend gliedert sich dieses Buch in mehrere Teile, wobei vor allem die Phasen rund um die Implementierung behandelt werden. Teil I adressiert die Phase »Analyse und Design«. Teil II und Teil III stellen diverse Themen zur Phase »Implementierung« vor. Abschließend wird im Teil IV die Phase »Qualitätssicherung und Test« beschrieben. Folgende Aufzählung konkretisiert die dort vorgestellten Themen:

- **Teil I »Java-Grundlagen, Analyse und Design«** – Teil I beginnt in Kapitel 2 mit der Vorstellung einer sinnvoll ausgestatteten Arbeitsumgebung, die beim Entwickeln von professionellen Programmen hilft und es ermöglicht, gute Resultate produzieren zu können. Anschließend wird in Kapitel 3 das Thema objektorientiertes Design beschrieben. Als Notation wird dabei die UML verwendet, die in Anhang A vorgestellt wird. Damit sind die Grundlagen für einen professionellen, objektorientierten Softwareentwurf gelegt und die Vorbereitungen für den Start mit dem Implementieren getroffen. Kapitel 4 stellt einige wichtige Java-Sprachelemente vor, die zum Verständnis der Beispiele in den folgenden Kapiteln notwendig sind.
- **Teil II »Bausteine stabiler Java-Applikationen«** – Teil II behandelt diverse Themen aus der Implementierungsphase. Hier werden wichtige Kenntnisse fundamentaler Java-APIs vermittelt, aber auch fortgeschrittene Java-Techniken behandelt. Zunächst stellt Kapitel 5 das Thema Collections vor, um eine effiziente Wahl von Datenstrukturen zur Speicherung und Verwaltung von Daten zu ermöglichen. In

Kapitel 6 beschäftigen wir uns mit der Erstellung wiederverwendbarer Softwarebausteine. Es ist wichtig, das Rad nicht immer neu zu erfinden, sondern auf einer stabilen Basis aufzubauen. Ein weiterer wichtiger Baustein beim professionellen Programmieren ist Multithreading. Kapitel 7 gibt eine fundierte Einführung und stellt auch fortgeschrittenere Themen, wie das Java-Memory-Modell und die Parallelverarbeitung mit Thread-Pools, vor. Weitere fortgeschrittenere Themen, etwa Garbage Collection, Internationalisierung usw. werden in Kapitel 8 behandelt. Den Abschluss bildet Kapitel 9, das einige wichtige Neuerungen in JDK 7 beschreibt.

Nach der Lektüre dieser ersten beiden Teile sind Sie programmiertechnisch fit und bereit für das Schreiben eigener Anwendungen mit komplexeren Aufgabenstellungen. Auf dem Weg zu guter Software werden wir über das eine oder andere Problem stolpern. Wie Sie mögliche Probleme erkennen und beheben, ist Thema der folgenden Teile.

- **Teil III »Fallstricke und Lösungen im Praxisalltag«** – Teil III betrachtet in Kapitel 10 ausführlich mögliche Programmierprobleme, sogenannte »Bad Smells«. Diese werden analysiert und Lösungsmöglichkeiten dazu aufgezeigt. Diverse Umbaumaßnahmen werden in Kapitel 11 als Refactorings vorgestellt. Kapitel 12 rundet diesen Teil mit der Präsentation einiger für den Softwareentwurf wichtigen Lösungsideen, sogenannter Entwurfsmuster, ab. Diese sorgen zum einen dafür, ein Problem auf eine dokumentierte Art und Weise zu lösen, und zum anderen, Missverständnisse zu vermeiden, da die Entwickler eine eigene, gemeinsame Designsprache sprechen.
- **Teil IV »Qualitätssicherungsmaßnahmen«** – Qualitätssicherung ist für gute Software elementar wichtig und wird in Teil IV vorgestellt. In den Bad Smells gewonnene Erkenntnisse werden in Kapitel 13 zu einem Regelwerk beim Programmieren, sogenannten Coding Conventions, zusammengefasst. Um neue Funktionalität und Programmänderungen abzusichern, schreiben wir Unit Tests. Nur durch eine breite Basis an Testfällen haben wir die Sicherheit, Änderungen ohne Nebenwirkungen auszuführen. Kapitel 14 geht detailliert darauf ein. Eine Qualitätskontrolle über Codereviews wird in Kapitel 15 thematisiert. Zu einer guten Qualität gehören aber auch nicht funktionale Anforderungen. Diese betreffen unter anderem die Performance eines Programms. In Kapitel 16 stelle ich daher einige Techniken zur Performance-Steigerung vor.
- **Anhang** – Anhang A dient als Kurzreferenz und zum Verständnis für die eingesetzten UML-Diagramme. In Anhang B wird ein Überblick über verschiedene Vorgehensmodelle bei der Softwareentwicklung gegeben.

1.2.2 Kapitelübersicht

Nachdem Sie nun einen groben Überblick über die Struktur haben und wissen, was Sie in den einzelnen Teilen des Buchs erwartet, möchte ich die Themen der einzelnen Kapitel kurz vorstellen, um Ihnen einen möglichen Leseweg durch das Buch vorzuschlagen.

Abbildung 1-1 stellt Abhängigkeiten und mögliche Lesewege durch dieses Buch grafisch dar.

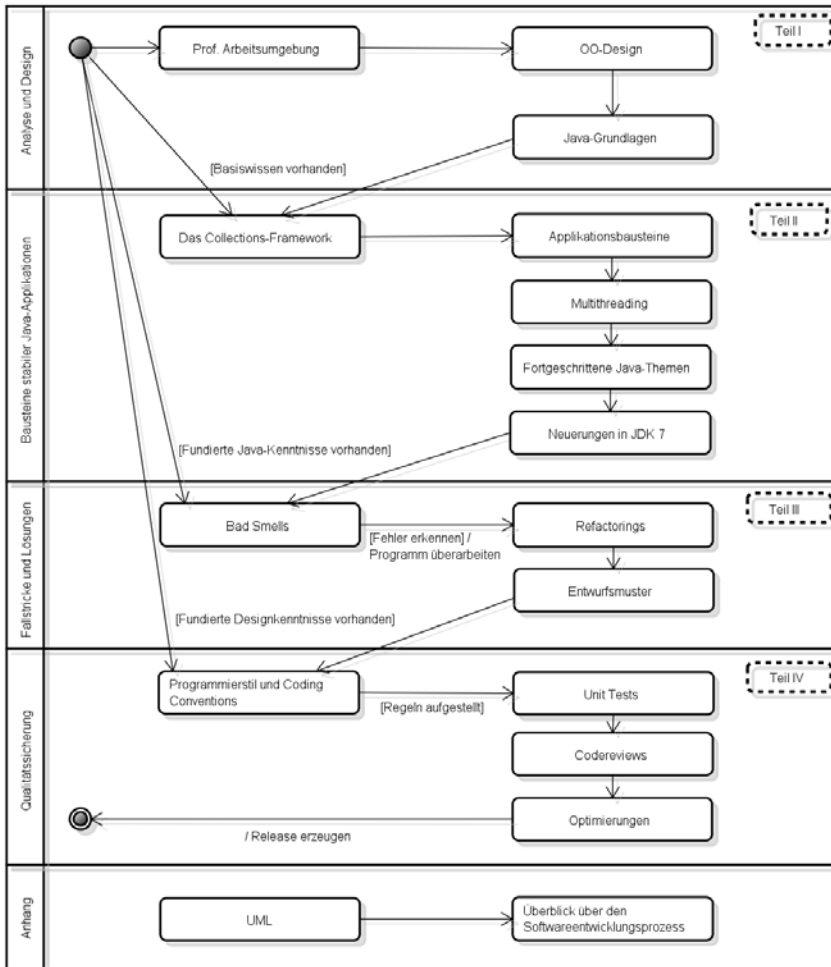


Abbildung 1-1 Mögliche Lesewege durch das Buch

Teil I – Java-Grundlagen, Analyse und Design

Dieser Teil legt die Grundlagen für einen guten Softwareentwurf, in dem sowohl auf objektorientiertes Design als auch auf eine produktive Arbeitsumgebung mit den richtigen Hilfsmitteln eingegangen wird. Teil I gliedert sich wie folgt:

Kapitel 2 – Professionelle Arbeitsumgebung Den Vorteil von Entwicklungsumgebungen und einer einheitlichen Projektstruktur beim Programmieren stellt dieses Kapitel vor. Weiterhin beschäftigen wir uns mit dem Thema Versionsverwaltung. Nur

durch den Einsatz einer solchen sind wir in der Lage, Änderungen gefahrlos wieder rückgängig zu machen und stabile Entwicklungsstände unter einem Namen für die Zukunft zu sichern. Zudem werfen wir einen ersten Blick auf das JUnit zum Erstellen von Unit Tests. Anschließend betrachten wir die Notwendigkeit, einen von der Entwicklungsumgebung unabhängigen Build durchführen zu können. Dazu werden wir das Tool Ant einsetzen.

Kapitel 3 – OO-Design Dieses Kapitel gibt einen Einblick in den objektorientierten Entwurf und stellt zunächst kurz die Grundbegriffe der Objektorientierung (kurz OO) vor, um anschließend das Thema Vererbung und Polymorphie zu diskutieren. Zusätzlich betrachten wir kritisch das Thema Vererbung. Neben einigen grundlegenden OO-Techniken werden im Verlauf des Kapitels auch einige fortgeschrittenere OO-Techniken vermittelt. Den Abschluss bildet eine Einführung in das Thema Generics und eine Betrachtung einer Kombination mit Vererbung.

Kapitel 4 – Java-Grundlagen Dieses Kapitel behandelt diverse Bereiche des Java-Basiswissens, die in der Praxis relevant sind und die man sicher anwenden können sollte. Unter anderem werden Kenntnisse über die Klasse `Object` sowie die Wrapper-Klassen der primitiven Typen vermittelt. Weiterhin werden die Sprachmöglichkeiten von Interfaces und inneren Klassen vorgestellt. Außerdem wird das Thema Ein- und Ausgabe in Dateien behandelt. Abgerundet wird das Kapitel durch eine Diskussion zur Behandlung von Fehlern durch Exceptions und Assertions.

Teil II – Bausteine stabiler Java-Applikationen

Der zweite Teil beschäftigt sich mit Komponenten und Bausteinen stabiler Java-Applikationen. Es werden diverse Themen vorgestellt, die als Bausteine für Applikationen dienen. Teil II gliedert sich wie folgt:

Kapitel 5 – Das Collections-Framework Keine komplexere Anwendung kommt ohne den Einsatz von Datenstrukturen aus. Glücklicherweise existieren in Java bereits die gebräuchlichsten Datenstrukturen wie Listen, Mengen und Schlüssel-Wert-Abbildungen. In diesem Kapitel lernen wir die Vorzüge, aber auch die Beschränkungen der einzelnen Datenstrukturen kennen und können später beim Entwurf eigener Programme die jeweils geeignete Datenstruktur für eine Problemstellung wählen.

Kapitel 6 – Applikationsbausteine Applikationen basierend auf funktionierenden Teilen zu konstruieren statt diese immer wieder von Grund auf neu zu bauen, macht den Softwareentwurf zu einer Ingenieurdisziplin. Man verhindert so, das Rad ständig neu zu erfinden. In diesem Kapitel werden wiederverwendbare Bausteine für eigene Applikationen konstruiert. Ich behandle die Themen Wertebereichs- und Parameterprüfungen, Konfigurationsverwaltung und Auswertung von Kommandozeilenparametern. Zudem bespreche ich das Thema Logging und Konfiguration von `log4j`.

Kapitel 7 – Multithreading Komplexe Aufgaben und Berechnungen lassen sich zum Teil gut in voneinander unabhängige Teilaufgaben untergliedern, die parallel abgearbeitet werden können. Dieses Kapitel beschreibt einerseits, wie man solche Berechnungen mit Threads durchführt, geht andererseits aber auch auf mögliche Probleme bei der Zusammenarbeit mehrerer Threads ein. Dabei spielen das Java-Memory-Modell und verschiedene Arten der Synchronisation eine wichtige Rolle. Nachdem die Grundbausteine erklärt wurden, werden die mit JDK 5 eingeführten Concurrency Utilities und die sich daraus ergebenden Erleichterungen bei der Programmierung von nebenläufigen Anwendungen besprochen.

Kapitel 8 – Fortgeschrittene Java-Themen In diesem Kapitel werden einige fortgeschrittene Java-Themen behandelt. Ich stelle das Thema Speichermanagement und Garbage Collection vor. Weiterhin gehe ich im Rahmen der Internationalisierung auf die Formatierung verschiedener Zeit- und Datumsformate, aber auch auf die Übersetzung von Benutzeroberflächen ein. Gute Reaktionszeiten in GUIs erhält man durch den Einsatz von Multithreading. Hierbei gibt es in Kombination mit Swing einige Besonderheiten zu beachten.

Kapitel 9 – Neuerungen in JDK 7 Zum Abschluss von Teil II stelle ich einige Neuerungen des JDK 7 anhand kleiner Beispielprogramme kurz vor.

Teil III – Fallstricke und Lösungen im Praxisalltag

Der dritte Teil beschreibt Probleme aus dem Praxisalltag und mögliche Lösungen. Dies wird anhand von Beispielen detailliert analysiert. Dadurch wird ein tieferes Verständnis für einen guten Softwareentwurf erlangt. Dieser Teil behandelt folgende Themen:

Kapitel 10 – Bad Smells Dieses Kapitel beschäftigt sich ausgiebig mit diversen Fallstricken beim Softwareentwurf. Einige in der Praxis immer wieder zu beobachtende Fehler werden anhand von Beispielen ausführlich analysiert. Es spart Zeit, Ärger und Frustration, diese Irrwege zu kennen. Zudem verhindert dies, die gleichen Fehler selbst zu machen und sich dadurch dann in Problemsituationen wiederzufinden. Entdeckt man jedoch in eigenen Programmen einen der hier vorgestellten Bad Smells, so werden Tipps und Tricks zur Korrektur gegeben.

Kapitel 11 – Refactorings Die Umgestaltung existierenden Sourcecodes ohne eine Änderung des Programmverhaltens wird als Refactoring bezeichnet. In diesem Kapitel stelle ich verschiedene Refactorings vor, um den Sourcecode zu verbessern. Grundlage für Refactorings ist einerseits der Einsatz einer Versionsverwaltung, sodass man jederzeit wieder auf einen definierten Stand zurückgehen kann, falls etwas schief läuft, und andererseits eine solide Basis an Unit Tests, um auszuschließen, dass die vorgenommenen Änderungen zu Problemen führen.

Kapitel 12 – Entwurfsmuster Dieses Kapitel stellt Lösungen und Ideen zu allgemeinen Entwurfsproblemen, sogenannte Entwurfsmuster, vor. Dabei lernen wir die in der Praxis gebräuchlichsten Muster u. a. SINGLETON, FABRIKMETHODE, KOMPOSITUM, ADAPTER, TEMPLATE-METHODE kennen. Entwurfsmuster helfen, Designs und die zugrunde liegenden Entscheidungen zu dokumentieren.

Teil IV – Qualitätssicherungsmaßnahmen

Teil IV nennt einige Methoden, die entstehende Software besser zu machen. Dabei betrachtet man beim Testen einer Applikation häufig lediglich die nach außen sichtbare, »gefühlte« Qualität. Diese steigt in der Regel proportional zu der inneren, softwaretechnischen Qualität an, die durch Unit Tests sowie Codierungsregeln und Codereviews verbessert werden kann. Dieser Teil behandelt hauptsächlich die innere Qualität und stellt folgende Themen vor:

Kapitel 13 – Programmierstil und Coding Conventions Probleme können nicht nur durch Fehler im Design oder der Implementierung verursacht werden, sondern bereits durch Missachten einiger Grundsätze provoziert werden. Das Risiko für Fehler lässt sich durch die Einhaltung einiger grundlegender Programmierregeln verringern. Zur Automatisierung der Überprüfung der Einhaltung werden verschiedene Tools vorgestellt.

Kapitel 14 – Unit Tests Dieses Kapitel beschreibt das Thema Testen und den Einfluss auf die Softwarequalität. Ohne Tests erreicht man normalerweise nur eine unterdurchschnittliche Qualität. Testen wird in diesem Kapitel anhand von Beispielen als entwicklungsbegleitende Tätigkeit motiviert, wie dies von den aktuellen Vorgehensmodellen (vgl. Anhang B) propagiert wird. Dieses Kapitel zeigt des Weiteren einige Tipps und Kniffe und geht auch auf fortgeschrittenere Unit-Test-Themen wie Multithreading, Mocks und Stubs ein.

Kapitel 15 – Codereviews In diesem Kapitel wird das Thema Codereview vorgestellt. Zunächst betrachten wir, was Codereviews überhaupt sind, und diskutieren einige Probleme, die sich bei der Durchführung ergeben können, und welche Lösungsmöglichkeiten existieren. Eine kurze Vorstellung einiger Tools sowie eine Checkliste, die als Ausgangsbasis für eigene Codereviews dienen kann, runden das Kapitel ab.

Kapitel 16 – Optimierungen Dieses Kapitel gibt einen Einblick in das Thema Optimierungen und stellt verschiedenste Optimierungstechniken vor. Der Einfluss verwendeter Algorithmen und Datenstrukturen sowie deren Auswirkungen auf die Performance werden untersucht. Zudem werden einige Techniken wie Pufferung, Objekt-Caching sowie Lazy Initialization besprochen und bewertet.

Teil V – Anhang

Anhang A – Einführung in die UML In diesem Kapitel wird die UML so weit vorgestellt, wie dies für den Gebrauch im praktischen Softwareentwurf und für das Verständnis der im Buch dargestellten Informationen notwendig ist. Während es früher viele unterschiedliche Entwurfsdialekte gab, hat sich im Laufe der letzten Jahre die UML als Beschreibung für den Entwurf von Software durchgesetzt.

Anhang B – Überblick über den Softwareentwicklungsprozess Die Abläufe bei der Softwareentwicklung können durch Vorgehensmodelle beschrieben werden. Dieses Kapitel stellt deren klassische Vertreter Wasserfall- und V-Modell sowie die zwei bekanntesten neueren Vorgehensmodelle »Extreme Programming (XP)« und »Test-Driven Development (TDD)« kurz vor.

1.3 Konventionen

Verwendete Zeichensätze

Im gesamten Text gelten folgende Konventionen bzgl. der Schriftart: Der normale Text erscheint in der vorliegenden Schriftart. Dabei werden wichtige Textpassagen *kursiv* oder *kursiv und fett* markiert. Englische Fachbegriffe werden eingedeutscht groß geschrieben. Zusammensetzungen aus englischen und deutschen (oder eingedeutschten) Begriffen werden mit Bindestrich verbunden, z. B. Plugin-Manager. Namen von Refactorings, Bad Smells sowie Entwurfsmustern werden bei ihrer Verwendung in KAPITÄLCHEN dargestellt. Sourcecode-Listings sind in der Schrift `courier` gesetzt, um zu verdeutlichen, dass dieser Text einen Ausschnitt aus einem realen Java-Programm darstellt. Auch im normalen Text werden Klassen, Methoden, Konstanten und Übergabeparameter in dieser Schriftart dargestellt.

Verwendete Abkürzungen

Im Buch verwende ich die in Tabelle 1-1 aufgelisteten Abkürzungen. Weitere Abkürzungen werden im laufenden Text in Klammern nach ihrer ersten Definition aufgeführt und anschließend bei Bedarf genutzt.

Verwendete Java-Umgebungen und -Versionen

Wir werden uns in diesem Buch vorwiegend mit Beispielen aus dem Bereich der Java Standard Edition (Java SE) auseinandersetzen. Wo dies sinnvoll ist, werde ich auf einige Ideen und Themen aus dem Bereich Java Enterprise Edition (Java EE oder JEE) eingehen. Diese ist eine Erweiterung der Standard Edition um die Möglichkeit, Client-Server-Systeme und größere, verteilte Applikationen zu entwickeln.

Diese Editionen existieren in verschiedenen Versionen. Die diesem Buch zugrunde liegende Version der Java Standard Edition ist Version 6, die voraussichtlich im Sommer

Table 1-1 *Verwendete Abkürzungen*

Abkürzung	Bedeutung
JDK	Java Development Kit
JLS	Java Language Specification
JRE	Java Runtime Environment
JSR	Java Specification Request
JVM	Java Virtual Machine
OO	Objektorientierung
TDD	Test-Driven Development
UML	Unified Modeling Language
XP	Extreme Programming
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
GUI/UI	(Graphical) User Interface
IDE	Integrated Development Environment
XML	Extensible Markup Language

2011 durch Version 7 abgelöst wird. Die meisten Beispiele basieren daher auf Java SE 6. Sehr verbreitet ist immer noch die Version 5. Tatsächlich wird in größeren Projekten zum Teil sogar noch Version 1.4 eingesetzt. Wo dies sinnvoll erscheint, werden daher einige Techniken kurz vorgestellt, die die Pflege älteren Sourcecodes erleichtern. Ältere Versionen als 1.4 sind kaum mehr im Einsatz und werden daher in diesem Buch nicht betrachtet.

Verwendete Klassen aus dem JDK

Werden Klassen des JDKs zum ersten Mal im Text erwähnt, so wird deren voll qualifizierter Name, d. h. inklusive der Package-Struktur, angegeben: Für die Klasse `String` würde dann etwa `java.lang.String` notiert. Dies erleichtert eine Orientierung und ein Auffinden im JDK. Dies gilt insbesondere, da in den Listings nur selten `import`-Anweisungen abgebildet werden. Im nachfolgenden Text wird zur besseren Lesbarkeit auf diese Angabe verzichtet und nur der Klassenname genannt.

Im Text beschriebene Methodenaufrufe enthalten in der Regel die Typen der Übergabeparameter, etwa `substring(int, int)`. Sind die Parameter in einem Kontext nicht entscheidend, wird auf deren Angabe aus Gründen der besseren Lesbarkeit verzichtet.

Sourcecode und ausführbare Programme

Um den Rahmen des Buchs nicht zu sprengen, stellen die abgebildeten Programm-listing häufig nur Ausschnitte aus lauffähigen Programmen dar. Weil dies ein Buch zum Mitmachen ist, existieren für viele Programme Ant-Targets (die lernen wir später in Kapitel 2 kennen). Deren Name wird in Kapitälchenschrift, etwa `LOCALEEXAMPLE`, angegeben.

Der Sourcecode kann auf der Webseite www.dpunkt.de/java-profi heruntergeladen werden. Dort findet sich auch ein Eclipse-Projekt, das importiert werden kann. Damit stehen dann auch alle Ant-Targets zur Verfügung.

1.4 Danksagungen

Ein Fachbuch zu schreiben ist eine schöne, aber arbeitsreiche und langwierige Aufgabe. Alleine kann man eine solche Aufgabe nicht bewältigen, daher möchte ich mich an dieser Stelle bei allen bedanken, die direkt oder indirekt zum Entstehen und Verbessern des Buchs beigetragen haben.

Zu meiner Zeit als Senior-Softwareentwickler und später als Softwarearchitekt bei der Heidelberger Druckmaschinen AG in Kiel ist bei den Vorbereitungen zu Codereview-Meetings und der Ausarbeitung von Fortbildungsvorträgen zum ersten Mal der Gedanke an ein solches Buch entstanden. Ich möchte all meinen damaligen Kollegen danken, die an diesen Meetings teilgenommen haben. Als Veranstalter und Vortragender lernt man immer wieder neue Details, die man vorher nicht bedacht hatte oder kannte. Dietrich Mucha und Reinhard Pupkes danke ich für ihre Korrekturen und Anmerkungen, die gemeinsamen Erfahrungen beim Ausarbeiten von Coding Conventions und Codereviews sowie die nette Zeit bei Pair Programming Sessions. Die Zusammenarbeit mit Tim Bötzmeyer hat mir viel Freude bereitet. Unsere langen, interessanten Diskussionen über Java und die Fallstricke beim OO-Design haben mir diverse neue Einblicke verschafft.

Auch einige Kollegen bei meinem jetzigen Arbeitgeber der IVU Traffic Technologies AG in Aachen haben sich die Zeit genommen, mein Manuskript zu lesen und mich mit Korrekturen und Anregungen zu unterstützen. Unter anderem danke ich Rudolf Braun, Christian Gehrman, Peter Kehren, Felix Korb und Roland Schmitt-Hartmann für den einen oder anderen Hinweis und Tipp. Sie haben mich an einigen Stellen mit diversen guten Ideen und Anmerkungen dazu gebracht, den Text weiter zu verbessern.

Mein spezieller Dank gilt Merten Driemeyer, der sich sehr gründlich mit frühen Entwürfen des Manuskripts beschäftigt und mir an diversen Stellen fachliche und sprachliche Tipps gegeben hat. Gleiches gilt für Dr. Iris Rottländer, die sowohl formal als auch inhaltlich an vielen Stellen durch ihre Anmerkungen zu einer Verbesserung des Textes gesorgt hat. Auch Dr. Carsten Kern und Andreas Schöneck haben gute Hinweise gegeben und einige verbliebene kleinere Fehler aufgedeckt. Last but not least haben die Anmerkungen von Dr. Clemens Gugenberger und unsere nachfolgenden Diskussionen einigen Kapiteln den letzten Feinschliff gegeben. Gleiches gilt für Stefan Bartels, der

mich immer wieder durch gute Anmerkungen unterstützt und damit zur Verständlichkeit des Textes beigetragen hat. Alle sechs haben mir entscheidend geholfen, inhaltlich für mehr Stringenz und Klarheit zu sorgen.

Mein größter Dank geht an Andreas Bubolz, der mich immer wieder unterstützt und enorm viel Zeit und Mühe investiert hat. Als Korrekturleser und Sparringspartner in vielen Diskussionen hat er diverse Unstimmigkeiten im Text aufgedeckt und damit dafür gesorgt, dass so wenig Fehler wie möglich im Text verblieben sind.

Außerdem danke ich meiner Frau Lilija Kurban für ihr Verständnis, für die zeitlichen Entbehrungen und die Unterstützung beim Ausarbeiten des Buchs.

Abschließend möchte ich mich beim gesamten Team des dpunkt.verlags bedanken, insbesondere Dr. Michael Barabas, der dieses Buchprojekt von der Idee bis zur Fertigstellung ermöglicht und gefördert hat. Weiterhin danke ich Vanessa Wittmer für ihre Unterstützung in organisatorischen Dingen. Auch Torsten Horns fachliche Durchsicht und das Copy Editing von Ursula Zimpfer haben mir sehr geholfen. Danke dafür!

Anregungen und Kritik

Ein Buch mit einigen Hundert Seiten wird immer irgendwelche Mängel und Fehler enthalten. Allerdings hoffe ich, die größten davon bereits entfernt zu haben. Nochmals vielen Dank an meine Korrekturleser, die dazu einen wertvollen Beitrag geleistet haben. Wenn sich doch noch Fehler – ob inhaltlicher oder formaler Art – eingeschlichen haben sollten und Ihnen auffallen, so zögern Sie nicht, mir diese mitzuteilen. Für Anregungen, Verbesserungsvorschläge und Kritik bin ich dankbar. Sie erreichen mich per Mail unter:

`michael_inden@hotmail.com`