

Unit Testing in Java

Johannes Link

With Contributions by Peter Fröhlich

Unit Testing in Java

How Tests Drive the Code

Johannes Link

With Contributions by Peter Fröhlich



MORGAN KAUFMANN PUBLISHERS

AN IMPRINT OF ELSEVIER SCIENCE

AMSTERDAM BOSTON LONDON NEW YORK
OXFORD PARIS SAN DIEGO SAN FRANCISCO
SINGAPORE SYDNEY TOKYO

Senior Editor	Tim Cox
Publishing Services Manager	Edward Wade
Editorial Coordinator	Stacie Pierce, Richard Camp
English translation	Angelika Shafir
Project Management	Matrix Productions, Inc.
Cover Design	Frances Baca
Cover Image	Photodisc Collection/Getty Images
Text Design	Rebecca Evans
Composition	Nancy Logan
Illustration	Dartmouth Publishing, Inc.
Copy Editor	Yoni Overton
Proofreader	Dan Young
Indexer	Edwin Durbin
Interior printer	The Maple-Vail Book Manufacturing Group
Cover printer	Phoenix Color Corporation

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Morgan Kaufmann Publishers
 An Imprint of Elsevier Science
 340 Pine Street, Sixth Floor
 San Francisco, CA 94104-3205
www.mkp.com

© 2002 by dpunkt.verlag GmbH, Heidelberg, Germany.
 Title of German original: Unit Tests Mit Java

English translation © 2003 by Elsevier Science (USA)
 All rights reserved.
 Printed in the United States of America

07 06 05 04 03 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, or otherwise—without the prior written permission of the publisher.

Library of Congress Cataloging-in-Publication Data
 Link, Johannes.

Unit testing in Java : how tests drive the code / by Johannes Link ; with contributions by Peter Fröhlich.

p. cm.

Includes bibliographical references and index.

ISBN 1-55860-868-0

1. Computer software—Testing. 2. Java (Computer program language)

I. Fröhlich, Peter. II. Title.

QA76.76.T48L55 2003

005.1'4—dc21

Morgan Kaufmann ISBN: 1-55860-868-0

dPunkt ISBN: 3-89864-150-3

This book is printed on acid-free paper.

Foreword

Erich Gamma

As a professional software developer, I want to develop software as fast as possible, as well as possible, and as stress-free as possible. Automated unit tests help to bring me closer to this goal. They are a small investment, which help me get confidence in the code I produce and maintain later on. When I don't have automated tests, I have to fall back on manual testing. However, manual tests cannot be automatically repeated. Consequently, the stress increases, particularly when they have to be done under time pressure, which isn't, of course, the exception. At the push of a button you can determine at any given time whether or not the last change impacts the fitness of your software. You can do this today, tomorrow, or any time in the future, regardless of whether or not a deadline is knocking at your door.

This book by Johannes Link and contributor Peter Fröhlich is a practical introduction to using automated unit tests and the test-first approach in your day-to-day software development. The automation framework used in the book is JUnit. It is a small and simple framework for creating and managing tests. However, more is needed for successful development with unit tests. In fact, a developer has to be familiar with many different testing techniques, in particular when unit tests have to be created in the context of databases or distributed applications based on application servers. This book sheds light on these problems and is a highly welcome contribution to the field of automated unit testing.

JUnit itself was also developed with automated unit tests and the test-first approach, as explained in this book. In fact, the techniques were even used under challenging conditions, such as while fighting jet lag or during

electric power failures in alpine cabins. Still, the techniques have proven themselves every time. I hope that, thanks to this book, you will become test infected and that in the future you will always be able to give a positive answer to the classical unit test control question, “Where are the unit tests?”

Erich Gamma
Co-author of JUnit
Technical Director, Object Technology International

Foreword

Frank Westphal

Do you remember your first programming experience? I don't mean the details of the computer or language used; I mean how did you feel?

I remember typing in a few statements from the programming handbook, eager to see the program run. It was amazing to watch how the code sprang to life. Within a few hours I had grown the book example into what seemed an impressive program. I had made additions here and there and after every change I would rerun the program to see how it was doing. In the evening I showed it to my parents. They could tell by the look in my eyes how proud I was.

How things have changed. I enjoy programming more than ever, but now and then I realize that some of the original fun is gone. It's in these moments that I reflect back on my first experience. Why can't programming always be like that?

Actually, it was in one of those moments that I came across the techniques described in this book. Automated tests and continuous refactoring applied in tandem brought me a bit closer to the beginning of my programming career. More often than not since then, I have been able to act like the only thing I have to do is write a few lines of code. But don't be fooled; these techniques are not only applicable to small programs. The larger the scale, the more valuable these techniques became to me.

I was glad when Johannes asked me to write a second foreword for two reasons. First, this book brings together the knowledge that a number of pioneering extreme programmers wish they had when they started applying test-first programming five years ago. If you follow down that route, you will invariably run into testing problems. Even though you are writing

your tests first, you will come to halt because you won't see how to test your code. That's natural. Actually, that's the perfect time to reflect. Or to pick up this book and read what Johannes tells us.

Second, I was going to write a book just like this one. However, when Johannes shared with me the first few chapters for review, I could see that it was well written and even covering a large suite of tools to support testing code that is usually hard, if not impossible, to test. I wish I had written this book. Therefore, writing an accompanying foreword is a great pleasure.

There is but one danger in reading this book. You might come away with the impression that it's all about techniques and tools. When in fact, it's all about you.

Test-infected programmers will tell you how the tests changed their relationship to the code. There is a certain fascination in seeing a few hundred tests passing and checking all the innards of your software. Indeed, sometimes you will find yourself pressing the run button a few more times, just for the extra kick that everything's working fine.

Frank Westphal

Independent trainer and consultant

Contents

Foreword	v
<i>Erich Gamma</i>	
Foreword	vii
<i>Frank Westphal</i>	
Preface	xvii

Part I

Basic Techniques

Chapter 1	Introduction	3
	1.1 Important Terms	5
	1.2 XP Testing	6
	Communication, Simplicity, Feedback, and Courage	7
	Pair Programming	7
	Incremental and Iterative Development	8
	Refactoring	8
	Test Types in XP	9
	XP or Not XP?	10
	1.3 Classic Testing	11
	1.4 Test-First Development—A Brief Definition	16
	1.5 Java Only—Or Other Coffee?	18
	1.6 Objectives of This Book	18
	1.7 Organization of This Book	19

- 1.8 Conventions in This Book 20
- 1.9 Web Site to This Book 21

Chapter 2

Automating Unit Tests 23

- 2.1 What Do We Want to Automate? 24
- 2.2 Requirements for an Automation Framework 25
- 2.3 JUnit 27
 - Installing and Running Tests 28
 - Creating Test Classes 30
 - Fixtures 34
 - Creating Test Suites 36
- 2.4 Summary 37

Chapter 3

Basic Steps of the Test-First Approach 39

- 3.1 Step by Step 39
- 3.2 Dependencies 48
- 3.3 Organizing and Running Tests 57
 - Organizing Tests 57
 - Running Tests 62
- 3.4 Summary 63

Chapter 4

Test Ideas and Heuristics 65

- 4.1 Reworking Single Tests 66
- 4.2 Black and White Boxes 70
- 4.3 Testing the Typical Functionality 71
- 4.4 Threshold Values and Equivalence Classes 73
- 4.5 Error Cases and Exceptions 75
- 4.6 Object Interactions 81
- 4.7 Design by Contract 84
- 4.8 More Ideas to Find Test Cases 86
- 4.9 Refactoring Code and Tests 87
- 4.10 Summary 90

Chapter 5

The Inner Life of a Test Framework 91

- 5.1 Statics 91
- 5.2 The Life Cycle of a Test Suite 93
- 5.3 Project-Specific Expansions 95
- 5.4 Summary 96

Chapter 6

Dummy and Mock Objects for Independence**97**

- 6.1 Little Dummies 97
- 6.2 Weltering in Technical Terms 100
- 6.3 Big Dummies 100
- 6.4 Extending Our Mansion 107
- 6.5 Endoscopic Testing 108
- 6.6 Mock Objects from the Assembly Line 113
 - Mock Library 113
 - Mock Generators 114
 - Mock Objects the Easy Way 114
- 6.7 Testing Threshold Values and Exceptions 116
- 6.8 How Does the Test Get to the Mock? 119
- 6.9 Evil Singletons 122
- 6.10 Lightweight and Heavyweight Mocks 124
- 6.11 File Dummies 129
- 6.12 More Typical Mock Objects 133
- 6.13 External Components 134
- 6.14 The Pros and Cons 137
 - Heuristics for the Use of Mocks 140
- 6.15 Summary 141

Chapter 7

Inheritance and Polymorphism**143**

- 7.1 Inheritance 143
 - Well-Shaped Inheritance Hierarchies 143
 - Reusing Superclass Tests 146
 - Test Class Hierarchies by Refactoring 151
 - Testing Interfaces 152
 - Testing Abstract Classes 155
- 7.2 Polymorphism 155
- 7.3 Summary 160

Chapter 8

How Much Is Enough?**161**

- 8.1 The XP Rule 162
- 8.2 Clear Answers to Clear Questions 163
 - Tests per Class 164
 - Getters and Setters 164
 - Non-Public Object Properties 164
 - Complex Interaction Tests 166
 - Testing the Tests 167

- 8.3 Test Coverage 167
 - Specification-Based Coverage 168
 - Code-Based Coverage 168
- 8.4 Summary 170

Part II

Advanced Topics

Chapter 9	Persistent Objects	173
	9.1 Abstract Persistence Interface 175	
	9.2 Persistent Dummy 178	
	9.3 Designing a Database Interface 181	
	Transactions 182	
	Ad Hoc Queries 184	
	Object-Centered Persistence 185	
	9.4 Testing the "Right" Persistence 187	
	Approaches for Test Data Consistency 191	
	Speeding Up the Test Suite 192	
	JDBC Mocks 193	
	Evolution of the Persistence Technology 195	
	9.5 Interaction between Persistence Layer and Client 196	
	9.6 Summary 198	
Chapter 10	Concurrent Programs	201
	10.1 Problems Using Threads 202	
	Nondeterminism 203	
	Target Objects 203	
	10.2 Testing Asynchronous Services 204	
	Service without Result 204	
	Service with Result 208	
	Expected Exceptions in Split-Off Threads 210	
	Unexpected Exceptions 212	
	10.3 Testing for Synchronization 212	
	Simple Test Cases 213	
	Concurrent Test Cases 214	
	Nondeterministic Test Cases 219	
	10.4 Summary 222	

Chapter 11

Distributed Applications 225

Distribution Mechanisms in Java 226

11.1 RMI 227

The Server 227

The Client 232

Summary of RMI 235

11.2 Enterprise JavaBeans 236

Just a Facade 237

Testing Inside the Container 238

EJBs and Simple Design 239

11.3 Summary 240

Chapter 12

Web Applications 241

12.1 Functional Tests 242

12.2 Testing on the Server 247

12.3 Testing with Dummies 250

12.4 Separating the Servlet API from the Servlet Logic 256

12.5 Testing the HTML Generation 259

Java Server Pages 260

JSP Custom Tags 260

Struts 261

12.6 Summary 261

Chapter 13

Graphical User Interfaces 263

13.1 The Direct Way 263

Brief Summary 283

Keeping the GUI Clear 285

13.2 Short Detours 286

JFCUnit 287

The AWT Robot 289

Other Tools 290

13.3 Summary 290

Chapter 14

The Role of Unit Tests in the Software Process 291

14.1 Activities in the Defined Software Process 292

Activities and Products 292

Construction Activities 294

Verification 295

- Validation 296
- Quality Assurance 297
- 14.2 Process Types and Testing Strategies 299
 - Sequential Models 300
 - Incremental Models 302
 - Evolutionary Models 303
 - Continuous Integration 305
- 14.3 Costs and Benefits of Automated Unit Tests 305
- 14.4 Commercial Process Models 307
 - Rational Unified Process 308
 - Extreme Programming 311
 - XP versus RUP 312
- 14.5 Will Automated Unit Tests Fit in My Process? 312

Chapter 15

Loose Ends and Opportunities 313

- 15.1 Unit Testing for Existing Software 314
 - Testing around Legacy Code 316
- 15.2 Introducing Unit Tests to the Development Team 317
 - The Craftsmanship Approach 318
 - The Organizational Approach 318
- 15.3 What's Missing? 320

Part III

Appendices

Appendix A

Notes to JUnit 325

- A.1 Frequently Asked Questions (FAQs) 325
 - How Do I Implement a Test Case for a Thrown Exception? 326
 - How Do I Organize My Test Case Classes? 326
 - How Do I Run Setup Code Once for All My Test Cases? 327
 - I Get a `ClassNotFoundException` When I Use `LoadingTestRunner`. What Can I Do? 327
 - Why Do I Get `Exception XYZ` When Using a Graphical Test Runner, But Not in the Textual Test Runner? 328
 - "assert" Has Been a Keyword Since JDK 1.4. Isn't There a Conflict with JUnit's `Assert` Method? 328
 - How Do I Best Integrate JUnit with My Favorite Development Environment? 328

- A.2 JUnit Expansions 329
 - JUnitX and XPTest 329
 - Daedalus JUnit Extensions 329
 - JFCUnit 330
 - JUnitPP 330
 - Mock Objects 330
 - MockMaker 330
 - EasyMock 331
 - JXUnit 331
 - JUnitHelp 331
 - Joshua 331
 - JDepend 331
 - JesTer 332
 - HttpUnit 332
 - JUnitEE 332
 - Canoo WebTest 333
 - Cactus 333
 - JUnitPerf 333
 - J2ME Unit 333
 - Test Mentor Java Edition 334

Appendix B

Unit Tests with Other Programming Languages 335

- B.1 Smalltalk 335
 - Creating Test Cases with SUnit 335
 - Differences to Java 336
 - Evaluation 337
- B.2 C++ 338
 - Installation 338
 - Setting Up a Test Project 339
 - Creating Test Cases 339
 - Running Your Tests 340
 - Overall Impression 343
- B.3 The Rest 343

Glossary 345

Bibliography and List of References 353

- Bibliography 353
- URLs 359
- Further Reading 362
 - Extreme Programming 362

Test-First and JUnit 363
Testing Object-Oriented Software 363
Miscellaneous 364

Index

365

Preface

When my German publisher told me that my book was going to be translated into English I was delighted: no work for lots of fame. This view was more than naive. Despite the translator's excellent job, the number of errors and omissions uncovered by the reviewers was quite a revelation.

So I had to take the challenge and rewrite parts of a book that I had already deemed finished and of high quality. What you hold in your hands now is maybe 70% translation and 30% new, updated, and improved material. And still I feel guilty for not having been able to seize all of the reviewers' suggestions.

As the book changed shape and went through heavy restructurings, my life did as well. Thanks to all who supported me in one process or another. These include the reviewers of both the German and the English versions: Frank Adler, Achim Bangert, Markus Barchfeld, Ekard Burger, Frank Cohen, Herbert Ehrlich, Eitan Farchi, Tammo Freese, Dierk König, Andreas Leidig, Erik Meade, Steve Metsker, Rainer Neumann, Christian Popp, Ilja Preuß, Stefan Roock, Michael Ruppert, Roland Sand, Martin Schneider, Thomas Singer, Andreas Schoolmann, Robert Wenner, Timothy Wall, and Frank Westphal; Angelika Shafir, who succeeded in translating not only the facts but also the spirit of the original text; Tim Cox and Stacie Pierce of Morgan Kaufmann Publishers, who value quality much higher than publication speed; Peter Fröhlich, co-author of the German version, who persevered through our discussions about language and style; all the people forming andrena objects, a more than suitable place to develop new ideas and to confront these ideas with reality; and Bettina and Jannek, who will hopefully help me fill many pages of our shared personal "book" with happiness and sadness.

