Beginning Java™ SE 6 Platform

From Novice to Professional

Jeff Friesen

Apress®

Beginning Java™ SE 6 Platform: From Novice to Professional

ISBN-13 (pbk): 978-1-59059-830-6

ISBN-10 (pbk): 1-59059-830-X

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Steve Anglin
Technical Reviewers: Sumit Pal, John Zukowski
Editorial Board: Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Jason Gilmore, Kevin Goff, Jonathan Hassell, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Richard Dal Porto
Copy Editor: Marilyn Smith
Assistant Production Director: Kari Brooks-Copony
Production Editor: Elizabeth Berry
Compositor: Gina Rexrode
Proofreader: April Eddy
Indexer: Becky Hornyak
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code/Download section.

To my parents and my good friend Amaury

Contents at a Glance

Preface xv
About the Author xvi
About the Technical Reviewers xvii
Acknowledgments xviii
Introduction xix

CHAPTER 1 Introducing Java SE 6 1
CHAPTER 2 Core Libraries 37
CHAPTER 3 GUI Toolkits: AWT 79
CHAPTER 4 GUI Toolkits: Swing 119
CHAPTER 5 Internationalization 153
CHAPTER 6 Java Database Connectivity 187
CHAPTER 7 Monitoring and Management 221
CHAPTER 8 Networking 253
CHAPTER 9 Scripting 281
CHAPTER 10 Security and Web Services 345
APPENDIX A New Annotation Types 381
APPENDIX B New and Improved Tools 389
APPENDIX C Performance Enhancements 409
APPENDIX D Test Your Understanding Answers 415
APPENDIX E A Preview of Java SE 7 455

INDEX 469

Contents

Preface xv
About the Author xvi
About the Technical Reviewers xvii
Acknowledgments xviii
Introduction xix

CHAPTER 1 Introducing Java SE 6 1

Name Change for This Java Edition 1
The Themes of Java SE 6 2
Overview of Java SE 6 4
Sampling of Java SE 6 New Features 5
A Trio of New Action Keys and a Method to Hide/Show Action Text 6
Clearing a ButtonGroup's Selection 12
Enhancements to Reflection 13
GroupLayout Layout Manager 14
Image I/O GIF Writer Plug-in 15
Incremental Improvements to String 16
LCD Text Support 17
NumberFormat and Rounding Modes 18
Improved File Infrastructure 20
Window Icon Images 21
Window Minimum Size 25
Interruptible I/O Switch for Solaris 25
ZIP and JAR Files 26
Ownerless Windows 26
Navigable Sets 29
Java SE 6, Update 1 and Update 2 34
Summary 35
Test Your Understanding 36

CHAPTER 2 Core Libraries 37

BitSet Enhancements 37
Compiler API 38
Access to the Compiler and Other Tools 39
The Standard File Manager 43
Compilation Task Futures 43
Diagnostic Information 45
String-Based Compilation 46
I/O Enhancements 49
Console I/O 49
Disk Free Space and Other Partition-Space Methods 52
File-Access Permissions Methods 54
Mathematics Enhancements 56
New and Improved Collections 57
More Collections Interfaces and Classes 57
More Utility Methods 64
New and Improved Concurrency 70
More Concurrent Interfaces and Classes 70
Ownable and Queued Long Synchronizers 72
Extension Mechanism and ServiceLoader API 73
Extension Mechanism 73
ServiceLoader API 73
Summary 76
Test Your Understanding 77

CHAPTER 3 GUI Toolkits: AWT 79

Desktop API 79
Dynamic Layout 87
Improved Support for Non-English Locale Input 91
New Modality Model and API 91
Splash Screen API 98
Making a Splash 98
Customizing the Splash Screen 99
System Tray API 103
Exploring the SystemTray and TrayIcon Classes 103
Quickly Launching Programs via the System Tray 110
XAWT Support on Solaris 117
Summary 117
Test Your Understanding 118

CHAPTER 4 GUI Toolkits: Swing 119

Arbitrary Components for JTabbedPane Tab Headers 119
Improved SpringLayout 125
Improved Swing Component Drag-and-Drop 126
JTable Sorting and Filtering 129
Sorting the Table's Rows 129
Filtering the Table's Rows 135
Look and Feel Enhancements 139
New SwingWorker 139
Text Component Printing 144
Summary 150
Test Your Understanding 151

CHAPTER 5 Internationalization 153

Japanese Imperial Era Calendar 153
Date Handling 153
Calendar Page Display 154
Locale-Sensitive Services 160
Service Provider Interface Classes 160
A New Currency for Java 162
New Locales 167
Normalizer API 167
ResourceBundle Enhancements 171
Taking Advantage of Cache Clearing 173
Taking Control of the getBundle() Methods 180
Summary 184
Test Your Understanding 185

CHAPTER 6 Java Database Connectivity 187

JDBC 4.0 187
Automatic Driver Loading 188
Enhanced BLOB and CLOB Support 189
Enhanced Connection Management 191
Enhanced Exception Handling 193
National Character Set Support 196
New Scalar Functions 197
SQL ROWID Data Type Support 199
SQL XML Data Type Support 201
Wrapper Pattern Support 202
Java DB 204
Java DB Installation and Configuration 205
Java DB Examples 207
Java DB Command-Line Tools 210
Play with the EMPLOYEE Database 214
Summary 219
Test Your Understanding 219

CHAPTER 7 Monitoring and Management 221

Dynamic Attach and the Attach API 221
Using the Attach API with the JMX Agent 224
Using the Attach API with Your Own Java-Based Agent 231
Improved Instrumentation API 236
Retransformation Support 238
Native Method Support 238
Support for Additional Instrumentation Classes 239
Improved JVM Tool Interface 240
Improved Management and JMX APIs 241
Management API Enhancements 242
JMX API Enhancements 243
JConsole GUI Makeover 244
JConsole Plug-ins and the JConsole API 245
A Basic Plug-in 246
Beyond the Basic Plug-in 249
Summary 251
Test Your Understanding 251

CHAPTER 8 Networking 253

CookieHandler Implementation 253
Internationalized Domain Names 257
An IDN Converter 259
A Better Browser 261
Lightweight HTTP Server 264
Network Parameters 267
SPNEGO HTTP Authentication 271
Challenge-Response Mechanism, Credentials, and Authentication Schemes 272
Basic Authentication Scheme and Authenticator Class 272
Digest Authentication 275
NTLM and Kerberos Authentication 276
GSS-API, SPNEGO, and the Negotiate Authentication Scheme . . . 276
Summary 278
Test Your Understanding 279

CHAPTER 9 Scripting 281

Scripting API Fundamentals 281
Obtaining Script Engines from Factories via the Script Engine Manager 284
Evaluating Scripts 290
Interacting with Java Classes and Interfaces from Scripts 292
Communicating with Scripts via Script Variables 294
Understanding Bindings and Scopes 296
Understanding Script Contexts 300
Generating Scripts from Macros 308
Compiling Scripts 309
Invoking Global, Object Member, and Interface-Implementing Functions 311
Playing with the Command-Line Script Shell 316
The Scripting API and JEditorPane 319
The Scripting API with JRuby and JavaFX Script 332
JRuby and the Scripting API 332
JavaFX Script and the Scripting API 336
Summary 342
Test Your Understanding 343

CHAPTER 10 Security and Web Services 345

Smart Card I/O API 345
XML Digital Signature APIs 349
Digital Signature Fundamentals 349
XML Signatures Standard 350
Java and the XML Signatures Standard 353
Web Services Stack 365
Creating and Testing Your Own Web Service 367
Accessing an Existing Web Service 371
Summary 377
Test Your Understanding 378

APPENDIX A New Annotation Types 381

Annotation Types for Annotation Processors 381
Common Annotations 1.0 382
More New Annotation Types 384

APPENDIX B New and Improved Tools 389

Basic Tools 389
Enhanced Java Archivist 389
Enhanced Java Language Compiler 391
Command-Line Script Shell 397
Java Monitoring and Management Console 399
Java Web Services Tools 400
Java Web Start 400
Security Tools 401
New keytool Options 401
New jarsigner Options 401
Troubleshooting Tools 402
Virtual Machine and Runtime Environment 407

APPENDIX C Performance Enhancements 409

A Fix for the Gray-Rect Problem 409
Better-Performing Image I/O 412
Faster Java Virtual Machine 413
Single-Threaded Rendering 414

APPENDIX D Test Your Understanding Answers 415

Chapter 1: Introducing Java SE 6 415
Chapter 2: Core Libraries 416
Chapter 3: GUI Toolkits: AWT 419
Chapter 4: GUI Toolkits: Swing 424
Chapter 5: Internationalization 425
Chapter 6: Java Database Connectivity 431
Chapter 7: Monitoring and Management 438
Chapter 8: Networking 443
Chapter 9: Scripting 445
Chapter 10: Security and Web Services 449

APPENDIX E A Preview of Java SE 7 455

Closures 455
JMX 2.0 and Web Services Connector for JMX Agents 457
More Scripting Languages and invokedynamic 458
New I/O: The Next Generation 458
Superpackages and the Java Module System 459
Swing Application Framework 460

INDEX 469

Preface

In late 2005, I started to explore Java SE 6 by writing a JavaWorld article titled "Start saddling up for Mustang" (`http://www.javaworld.com/javaworld/jw-01-2006/jw-0109-mustang.html`). This article investigated Console I/O, partition-space methods, the Splash Screen API, and the System Tray API.

In mid-2006, I wrote "Mustang (Java SE 6) Gallops into Town" (`http://www.informit.com/articles/article.asp?p=661371&rl=1`) for informit.com. This article continued my earlier Java SE 6 exploration by focusing on access permissions control methods, the Desktop API, programmatic access to network parameters, and table sorting and filtering.

In late 2006, I completed my article-based coverage of Java SE 6 by writing a trilogy of articles for informit.com: "Taming Mustang, Part 1: Collections API" (`http://www.informit.com/articles/article.asp?p=696620&rl=1`), "Taming Mustang, Part 2: Scripting API Tour" (`http://www.informit.com/articles/article.asp?p=696621&rl=1`), and "Taming Mustang, Part 3: A New Script Engine" (`http://www.informit.com/articles/article.asp?p=696622&rl=1`).

This book continues my exploration of Java SE 6.

About the Author

JEFF FRIESEN has been actively involved with Java since the late 1990s. Jeff has worked with Java in various companies, including a health-care–oriented consulting firm, where he created his own Java/C++ software for working with smart cards. Jeff has written about Java in numerous articles for JavaWorld.com, informit.com, and java.net, and has authored *Java 2 by Example, Second Edition* (Que Publishing). Jeff has also taught Java in university and college continuing education classes. He has a Bachelor of Science degree in mathematics and computer science from Brandon University in Brandon, Manitoba, Canada.

About the Technical Reviewers

SUMIT PAL has about 14 years of experience with software architecture, design, and development on a variety of platforms, including Java, J2EE. He has worked in the SQL Server Replication group while with Microsoft, and with Oracle's OLAP Server group while with Oracle. Apart from certifications such as IEEE-CSDP and J2EE Architect, Sumit has a Master of Science degree in Computer Science. Sumit has a keen interest in database internals, algorithms, and search engine technology. He currently works as an OLAP architect for LeapFrogRX. Sumit has invented some basic generalized algorithms to find divisibility between numbers, and also invented divisibility rules for prime numbers less than 100. Sumit has a fierce desire to work for Google some day.

JOHN ZUKOWSKI performs strategic Java consulting for JZ Ventures, Inc. He regularly contributes to Sun's monthly Tech Tips column and Java Technology Fundamentals newsletter. In addition, John monitors IBM's client-side Java programming forum at developerWorks. Since the beginning of Java time, John has authored ten books solo and contributed to several others. His best sellers include three editions each of the *Definitive Guide to Swing* (Apress) and *Mastering Java 2* (Sybex), and his latest, the predecessor to this book, *Java 6 Platform Revealed* (Apress).

Acknowledgments

I thank Steve Anglin for giving me the opportunity to continue my exploration of Java SE 6 via this book. I also thank Richard Dal Porto for guiding me through various aspects of the writing process. Thank you Sumit and John for your diligence in catching various flaws (including some embarrassing ones) that would otherwise have made it into this book. Finally, I thank Marilyn Smith, Elizabeth Berry, and April Eddy for making the book's content look good.

Introduction

Welcome to *Beginning Java SE 6 Platform*. Contrary to its title, this is not another beginner-oriented book on Java. You will not learn about classes, threads, file I/O, and other fundamental topics. If learning Java from scratch is your objective, you will need to find another book. But if you need to know (or if you just happen to be curious about) what makes Java SE 6 stand apart from its predecessors, this book is for you.

This book starts you on a journey of exploration into most of Java SE 6's new and improved features. Unfortunately, various constraints kept me from covering every feature, including the JavaBeans Activation Framework (<<sigh>>).

While you learn about these features, you'll also encounter exciting technologies, such as JRuby and JavaFX, and even catch a glimpse of Java SE 7. You'll also find numerous questions and exercises that challenge your understanding of Java SE 6, and numerous links to web resources for continuing this journey.

Beginning Java SE 6 Platform is a must-have resource if you want to quickly upgrade your skills. It is also the right choice if you need information about performance and other important topics before deciding if your company should upgrade to Java SE 6. This book will save you from wading through Java SE Development Kit (JDK) documentation and performing a lot of Internet searches.

Authors have idiosyncrasies; I am no different. For starters, although you'll often find links to various resources, I do not include links to entries in Sun's Bug Database. Rather than present individual links, I present bug identifiers and their names (Bug 6362451 "The string returned by toString() shows the bridge methods as having the volatile modificator," for example). If you want to find information about a bug, point your browser to `http://bugs.sun.com/bugdatabase/index.jsp`, enter the bug identifier in the appropriate field, and perform a search. In addition to the appropriate database entry appearing at the start of the search results, other results point you to related items that can enhance your understanding of a particular bug topic.

Other idiosyncrasies that you'll discover include my placing a `// filename.java` comment at the start of a source file (I forget the reason why I started to do this; old habits die hard), placing space characters between method names and their argument/parameter lists in source listings, importing everything from a package (`import java.awt.*;`, for example), limiting my comments in source listings, bolding certain parts of source listings to emphasize them, and adding the package name (unless the package is `java.lang`) to the first mention of a class or an interface in the text.

Who This Book Is For

This book assumes that you are a professional Java developer with a solid understanding of Java 2 Platform, Standard Edition 5 (J2SE 5). If you are new to Java, you'll probably feel overwhelmed by this book's content because it does not revisit basic Java concepts (such as classes and generics). It just is not possible to cover both the fundamentals and Java SE 6's new features in a single book.

For a version-agnostic treatment of Java and object-oriented fundamentals in general, refer to *Beginning Java Objects, Second Edition* (Apress, 2005; ISBN: 1-59059-457-6) by Jacquie Barker.

How This Book Is Structured

This book is organized into ten chapters and five appendixes. The first chapter introduces you to Java SE 6. The remaining chapters explore new and improved features in specific topic areas, in a tutorial style. The first three appendixes present additional features in a reference format. The penultimate appendix presents answers and solutions to the questions and exercises that are presented in Chapters 1 through 10. The final appendix gives you a preview of features that will most likely appear in Java SE 7. Here's a brief summary of the contents:

Chapter 1, Introducing Java SE 6: Every journey needs a beginning. Chapter 1 sets the stage for the remaining chapters by introducing you to Java SE 6. You'll learn the reason for the name change (it's not J2SE 6), the themes that define this release, and the big picture of what constitutes Java SE 6. You'll then get a taste of what is new and improved by exploring some Java SE 6 features not covered elsewhere in the book. Because Java SE 6 has evolved since build 105 (which is the build that I used to develop this book's code and examples), this chapter concludes with brief coverage of Java SE 6, update 1 and update 2.

Chapter 2, Core Libraries: Chapter 2 explores various core library topics. You'll learn about enhancements made to the `BitSet` class, the new Compiler API, I/O enhancements, mathematics enhancements, new and improved collections, new and improved concurrency, and the new ServiceLoader API. What are classpath wildcards? You'll find the answer in Chapter 2.

Chapter 3, GUI Toolkits: AWT: A lot of new stuff has been added to Java SE 6's Abstract Windowing Toolkit (or Abstract Window Toolkit, if you prefer). Chapter 3 explores the brand-new Desktop, Splash Screen, and System Tray APIs. It also looks at the new modality model and API. Various improvements have also been made to the existing infrastructure. This chapter briefly examines enhancements in the areas of dynamic layout, non-English locale input, and XAWT (the AWT for Solaris and Linux).

Chapter 4, GUI Toolkits: Swing: Not to be outdone, Swing has also benefited in Java SE 6. In Chapter 4, you'll learn how to add arbitrary components to `JTabbedPane`'s tab headers. You'll also examine the improvements in the SpringLayout layout manager and in the area of dragging and dropping Swing components. Then you'll play with the new `JTable` class features for sorting and filtering table contents, learn about enhancements to the Windows and GTK look and feels, and explore the new `SwingWorker` class. Finally, you'll discover how to print text components.

Chapter 5, Internationalization: Chapter 5 introduces you to the `Calendar` class's support for the Japanese Imperial Era calendar, the locale-sensitive services, new locales, the Normalizer API, and `ResourceBundle` enhancements. Among other things, you'll learn how the locale-sensitive services are used to introduce an appropriate currency provider for a new locale.

Chapter 6, Java Database Connectivity: This chapter has a "split personality." The first half focuses on new Java Database Connectivity (JDBC) features ranging from automatic driver loading to wrapper pattern support. The second half explores Java DB (also known as Apache Derby), which happens to be a pure-Java database management system (DBMS) bundled with JDK 6. If you are unfamiliar with Java DB/Derby, this chapter will quickly get you up to speed on using this technology. This chapter's "Test Your Understanding" section provides an example of going beyond this book by challenging you to describe how to get MySQL Connector/J 5.1 to support automatic driver loading.

Chapter 7, Monitoring and Management: Java SE 6 brings important changes and additions to the area of monitoring and management. Chapter 7 first presents dynamic attach and the new Attach API. The dynamic attach mechanism allows JConsole to connect to and start the Java Management Extensions (JMX) agent in a target virtual machine, and the Attach API allows JConsole and other Java applications to take advantage of this mechanism. After having some fun with this feature, you'll explore the improved Instrumentation API, JVM Tool Interface, and Management and JMX APIs. Moving on, you'll learn about the JConsole tool's improved graphical user interface (GUI). Finally, you'll explore the concept of JConsole plugins and examine the JConsole API.

Chapter 8, Networking: Chapter 8 focuses on Java SE 6's networking enhancements. To complement Java 5's introduction of the abstract `CookieHandler` class, Java SE 6 provides a concrete `CookieManager` subclass, which makes it easy to list a web site's cookies. After examining this topic, Chapter 8 focuses on internationalized domain names; you'll learn something interesting about `JEditorPane`'s `setPage()` methods. Then you'll be introduced to the new lightweight HTTP server and its API. (You'll

discover this server's usefulness in Chapter 10.) Next, you'll learn about network parameters. Developers of networked games will find one of the new network parameter methods described in this chapter especially helpful. Finally, the chapter introduces the topic of SPNEGO-based HTTP authentication.

Chapter 9, Scripting: Chapter 9 introduces both the new Scripting API and the experimental `jrunscript` tool. You'll learn how your applications can benefit from having access to JavaScript. This is one of my favorite chapters because it also discusses JRuby and JavaFX, but only from a Scripting API perspective.

Chapter 10, Security and Web Services: Chapter 10 is another "split-personality" chapter. It begins with a look at two new security features: the Smart Card I/O and XML Digital Signature APIs. Then it explores the new support for web services, via a web services stack and assorted tools.

Appendix A, New Annotation Types: Appendix A provides a reference on the new annotation types introduced by Java SE 6. These types are organized into three categories: annotation types supported by annotation processors, Common Annotations 1.0, and additional annotation types for the Java Architecture for XML Binding (JAXB), Java API for XML Web Services (JAX-WS), Java Web Service (JWS), JMX, and JavaBeans APIs.

Appendix B, New and Improved Tools: Appendix B provides a reference to changes made to existing tools and the introduction of new tools. This tool-related information is organized into the categories of basic tools, command-line script shell, monitoring and management console, web services tools, Java Web Start, security tools, and troubleshooting tools. This appendix also reviews many of the enhancements to the virtual machine and runtime environment. Additional enhancements related to virtual machine performance are discussed in Appendix C.

Appendix C, Performance Enhancements: In addition to robustness, Java SE 6's performance enhancements are a good reason to upgrade to this version. Appendix C provides a reference on some of these enhancements: a fix to the gray-rect problem (this is more than just a perceived problem with performance), better-performing Image I/O, faster HotSpot virtual machines, and single-threaded rendering.

Appendix D, Test Your Understanding Answers: Each of Chapters 1 through 10 ends with a "Test Your Understanding" section. Appendix D provides my answers to these questions and my solutions to these exercises. I recommend giving each question/exercise a good try before looking up its answer/solution in this appendix.

Appendix E, A Preview of Java SE 7: Java SE 7 (assuming that Sun does not change the naming convention) will probably debut in mid-to-late 2008. As the Java community's focus shifts from Java SE 6 to Java SE 7, you'll want to know what you can expect from this upcoming release. In Appendix E, I "polish my crystal ball" and give you a glimpse of what will most likely be included in Java SE 7. As with Java 5 (I refer to Java 5 instead of J2SE 5 throughout the book), you can expect some sort of language changes (closures, I predict). You can also expect new APIs, such as the Swing Application Framework. You'll explore these and other items in Appendix E.

Prerequisites

This book assumes that you are using Java SE 6 build 105 or higher. The book's content and code have been tested against build 105.

Downloading the Code

The sample code associated with this book is available from the Source Code/Download area of the Apress web site (`http://www.apress.com`). After you have downloaded and unzipped the file that contains this book's code, you'll discover a `build.xml` file. This file conveniently lets you use Apache Ant 1.6.5 (and probably higher versions as well) to build most of the code. You will also find a `README.txt` file that contains instructions for building the code with Ant.

Contacting the Author

Feel free to contact me about the content of this book, the downloadable code, or any other related topic, at `jeff@javajeff.mb.ca`. Also, please visit my web site at `http://javajeff.mb.ca`.

CHAPTER 1

Introducing Java SE 6

Java SE 6, the sixth generation of Java Standard Edition since version 1.0, officially arrived on December 11, 2006. This release offers many features that will benefit Java developers for years to come. This chapter introduces you to Java SE 6 and some of its features via the following topics:

- Name change for this Java edition
- Themes of Java SE 6
- Overview of Java SE 6
- Sampling of Java SE 6 new features
- Java SE 6, update 1 and update 2

Tip Meet the developers behind Java SE 6 by visiting the Planet JDK site (`http://planetjdk.org/`), which was created by Java SE Chief Engineer Mark Reinhold (see "Announcing planetjdk.org" at `http://weblogs.java.net/blog/mreinhold/archive/2005/11/announcing_plan.html`). You can learn a lot about Java SE 6 by reading the developers' blogs and articles. I present links to relevant blog and article entries throughout this book.

Name Change for This Java Edition

At different times during Java's 12-year history, Sun has introduced a new naming convention for its assorted Java editions, development kits, and runtime environments. For example, Java Development Kit (JDK) 1.2 became known as Java 2 Platform, Standard Edition 1.2 (J2SE 1.2). More recently, Sun announced that the fifth generation of its standard edition (since JDK 1.0) would be known as Java 2 Platform, Standard Edition 5.0 (J2SE 5.0), instead of the expected Java 2 Platform, Standard Edition 1.5.0 (J2SE 1.5.0).

The 5.0 is known as the external version number, and 1.5.0 is used as the internal version number.

Prior to releasing the latest generation, Sun's marketing team met with a group of its Java partners, and most agreed to simplify the Java 2 Platform's naming convention to build brand awareness. In addition to dropping the *2* from Java 2 Platform, Standard Edition, the "dot number" (the number following the period, as in 5.0) would be dropped, so that future updates to the Java platform would be noted as updates rather than dot numbers tacked onto the end of platform names. Hence, this latest Java release is known as *Java Platform, Standard Edition 6* (*Java SE 6*).

Similar to the 5.0 in J2SE 5.0 (which I refer to as Java 5 throughout this book), 6 is the external version number in the latest release. Also, 1.6.0 is the internal version number, which appears in the various places identified on Sun's Java SE 6, Platform Name and Version Numbers page (`http://java.sun.com/javase/6/webnotes/version-6.html`). This page also indicates that JDK (which now stands for Java SE Development Kit) continues to be the acronym for the development kit, and JRE continues to be the acronym for the Java Runtime Environment.

■ **Note** Jon Byous discusses the new naming convention in more detail via his "Building and Strengthening the Java Brand" article (`http://java.sun.com/developer/technicalArticles/JavaOne2005/naming.html`). Also, check out Sun's "New! Java Naming Gets a Birthday Present" article (`http://www.java.com/en/about/brand/naming.jsp`).

The Themes of Java SE 6

Java SE 6 was developed under Java Specification Request (JSR) 270 (`http://jcp.org/en/jsr/detail?id=270`), which presents the themes listed in this section. The themes are also mentioned in Sun's official press release on Java SE 6, "Sun Announces Revolutionary Version of Java Technology – Java Platform Standard Edition 6" (`http://www.sun.com/smi/Press/sunflash/2006-12/sunflash.20061211.1.xml`).

Compatibility and stability: Many members of the Java community have invested heavily in Java technology. Because it is important that their investments are preserved, effort has been expended to ensure that the vast majority of programs that ran on previous versions of the Java platform continue to run on the latest platform. A few programs may need to be tinkered with to get them to run, but these should be rare. Stability is just as important as compatibility. Many bugs have been fixed, and the HotSpot virtual machines and their associated runtime environments are even more stable in this release.

Diagnosability, monitoring, and management: Because Java is widely used for mission-critical enterprise applications that must be kept running, it is important to have support for remote monitoring, management, and diagnosis. To this end, Java SE 6 improves the existing Java Management Extensions (JMX) API and infrastructure, as well as JVM Tool Interface. For example, you now have the ability to monitor applications not started with a special monitoring flag (you can look inside any running application to see what is happening under the hood).

Ease of development: Java SE 6 simplifies a developer's life by providing new annotation types, such as `@MXBean` for defining your own MBeans; a scripting framework that you can use to leverage the advantages offered by JavaScript, Ruby, and other scripting languages; redesigned Java Database Connectivity (JDBC) that benefits from automatic driver loading; and other features.

Enterprise desktop: As developers encounter the limitations of browser-based thin clients, they are once again considering rich client applications. To facilitate the migration to rich client applications, Java SE 6 provides better integration with native desktop facilities (such as the system tray, access to the default web browser and other desktop helper applications, and splash screens), the ability to print the contents of text components, the ability to sort and filter table rows, font anti-aliasing so that text is more readable on liquid crystal display (LCD) screens, and more.

XML and web services: Java SE 6 provides significant enhancements in the area of XML; XML digital signatures and Streaming API for XML (StAX) are two examples. Although Java 5 was supposed to include a web services client stack, work on this feature could not be finished in time for Java 5's release. Fortunately, Java SE 6 includes this stack—hello, Web 2.0!

Transparency: According to JSR 270, "Transparency is new and reflects Sun's ongoing effort to evolve the J2SE platform in a more open and transparent manner." This is in response to the desire of many developers to participate more fully in the development of the next generation of Java. Because of the positive reception to Sun's "experiment in openness"—making Java 5 (Tiger) snapshot releases available to the public, which allowed developers to collaborate with Sun on fixing problems—Sun enhanced this experiment for Java SE 6. This transparency has fully evolved into Sun open-sourcing the JDK. Developers now have more influence on the features to be made available in the next generation of Java.

■ **Note** For more information about Java SE 6 transparency and open-sourcing, see Java SE Chief Engineer Mark Reinhold's "Mustang Snapshots: Another experiment in openness" blog entry (`http://weblogs.java.net/blog/mreinhold/archive/2004/11/index.html`) and the OpenJDK Community page (`http://community.java.net/openjdk/`).

Not every Java SE 6 feature is associated with a theme. For example, the class file specification update does not belong to any of the aforementioned themes. Also, not every theme corresponds to a set of features. For example, transparency reflects Sun's desire to be more open in how it interacts with the Java community while developing a platform specification and the associated reference implementation. Also, compatibility constrains how the platform evolves, because evolution is limited by the need to remain compatible with previous releases to support the existing base of Java software.

Overview of Java SE 6

Java SE 6 (which was formerly known by the code name Mustang during development) enhances the Java platform via improvements to the platform's performance and stability, by fixing assorted bugs, and even improvements to make graphical user interfaces (GUIs) look better (anti-aliasing LCD text is an example). Java SE 6 also enhances the Java platform by introducing a rich set of completely new features, some of which I've already mentioned. Many of these new features were developed by the various component JSRs of JSR 270, which serves as the "umbrella" JSR for Java SE 6:

- JSR 105: XML Digital Signature APIs (`http://jcp.org/en/jsr/detail?id=105`)
- JSR 199: Java Compiler API (`http://jcp.org/en/jsr/detail?id=199`)
- JSR 202: Java Class File Specification Update (`http://jcp.org/en/jsr/detail?id=202`)
- JSR 221: JDBC 4.0 API Specification (`http://jcp.org/en/jsr/detail?id=221`)
- JSR 222: Java Architecture for XML Binding (JAXB) 2.0 (`http://jcp.org/en/jsr/detail?id=222`)
- JSR 223: Scripting for the Java Platform (`http://jcp.org/en/jsr/detail?id=223`)
- JSR 224: Java API for XML-Based Web Services (JAX-WS) 2.0 (`http://jcp.org/en/jsr/detail?id=224`)
- JSR 268: Java Smart Card I/O API (`http://jcp.org/en/jsr/detail?id=268`)

- JSR 269: Pluggable Annotation Processing API (`http://jcp.org/en/jsr/detail?id=269`)

The one JSR specified in JSR 270's list of component JSRs that was not included in Java SE 6 is JSR 260: Javadoc Tag Technology Update (`http://jcp.org/en/jsr/detail?id=260`). Additional JSRs not specified in JSR 270's list, but that did make it into Java SE 6, are as follows:

- JSR 173: Streaming API for XML (`http://jcp.org/en/jsr/detail?id=173`)
- JSR 181: Web Services Metadata for the Java Platform (`http://jcp.org/en/jsr/detail?id=181`)
- JSR 250: Common Annotations for the Java Platform (`http://jcp.org/en/jsr/detail?id=250`)

Although these JSRs provide insight into what has been included in Java SE 6, "What's New in Java SE 6" (`http://java.sun.com/developer/technicalArticles/J2SE/Desktop/javase6/beta2.html`) offers a more complete picture. This article presents Danny Coward's "Top 10 Things You Need to Know" list of new Java SE 6 features (Danny Coward is the platform lead for Java SE), and Mark Reinhold's table of approved features. Of the table's listed features, internationalized resource identifiers (IRIs), the ability to highlight a `javax.swing.JTable`'s rows, and reflective access to parameter names did not make it into Java SE 6. IRIs, explained in RFC 3987: Internationalized Resource Identifiers (IRIs) (`http://www.ietf.org/rfc/rfc3987.txt`) were removed from the final release of Java SE 6 as part of `java.net.URI` being rolled back to the Java 5 version; see Bug 6394131 "Rollback URI class to Tiger version" in Sun's Bug Database").

■ **Note** The JDK 6 documentation's main page (`http://java.sun.com/javase/6/docs/`) presents a New Features and Enhancements link to the Features and Enhancements page (`http://java.sun.com/javase/6/webnotes/features.html`), which has more information about what is new and improved in Java SE 6.

Sampling of Java SE 6 New Features

As you will have noticed from the various feature references in the previous two sections, Java SE 6 has a lot to offer. This book explores most of Java SE 6's new and improved features, ranging from enhancements to the core libraries to a variety of performance enhancements. Before moving on, let's sample some of the features that set Java SE 6 apart from its predecessors.

A Trio of New Action Keys and a Method to Hide/Show Action Text

The `javax.swing.Action` interface extends the `java.awt.event.ActionListener` interface to bundle, in the same class, several component properties such as `toolTipText` and `icon` with common code. An instance of this class can be attached to multiple components (an Open menu item on a File menu and an Open button on a toolbar, for example), which then can be enabled/disabled from one place. Furthermore, selecting either component executes the common code. Java SE 6 lets you manipulate two new properties and a variation of `icon` via these new keys:

- `DISPLAYED_MNEMONIC_INDEX_KEY`: Identifies the index in the `text` property (accessed via the `NAME` key) where a mnemonic decoration should be rendered. This key corresponds to the new `displayedMnemonicIndex` property; the key's associated value is an `Integer` instance.

- `LARGE_ICON_KEY`: Identifies the `javax.swing.Icon` that appears on various kinds of Swing buttons, such as an instance of `javax.swing.JButton`. The `javax.swing.JMenuItem` subclasses, such as `javax.swing.JCheckBoxMenuItem`, use the `Icon` associated with the `SMALL_ICON` key. Unlike `LARGE_ICON_KEY`, there is no `SMALL_ICON_KEY` constant with a `_KEY` suffix.

- `SELECTED_KEY`: Initializes the selection state of a toggling component, such as an instance of `javax.swing.JCheckBox`, from an action and reflects this change in the component. This key corresponds to the new `selected` property; the key's associated value is a Boolean instance.

Java SE 6 also adds new action-related `public void setHideActionText(boolean hideActionText)` and `public boolean getHideActionText()` methods to the `javax.swing.AbstractButton` class. The former method sets the value of the `hideActionText` property, which determines whether (`true` passed to `hideActionText`) or not (`false` passed to `hideActionText`) a button displays an action's text; by default, a toolbar button does not display this text. The latter method returns this property's current setting. Listing 1-1 presents a notepad application that demonstrates these new action keys and methods.

Listing 1-1. *Notepad.java*

```
// Notepad.java

import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
```

```
import javax.swing.border.*;

public class Notepad extends JFrame
{
   private JTextArea document = new JTextArea (10, 40);

   public Notepad ()
   {
      super ("Notepad 1.0");
      setDefaultCloseOperation (EXIT_ON_CLOSE);

      JMenuBar menuBar = new JMenuBar ();

      JToolBar toolBar = new JToolBar ();

      JMenu menu = new JMenu ("File");
      menu.setMnemonic (KeyEvent.VK_F);

      Action newAction = new NewAction (document);
      menu.add (new JMenuItem (newAction));
      toolBar.add (newAction);

      // Java SE 6 introduces a setHideActionText() method to determine
      // whether or not a button displays text originating from an action. To
      // demonstrate this method, the code below makes it possible for a
      // toolbar button to display the action's text -- a toolbar button does
      // not display this text in its default state.

      JButton button = (JButton) toolBar.getComponentAtIndex (0);
      button.setHideActionText (false);

      menuBar.add (menu);

      menu = new JMenu ("View");
      menu.setMnemonic (KeyEvent.VK_V);

      Action statAction = new StatAction (this);
      menu.add (new JCheckBoxMenuItem (statAction));

      menuBar.add (menu);

      setJMenuBar (menuBar);
```

```
      getContentPane ().add (toolBar, BorderLayout.NORTH);
      getContentPane ().add (document, BorderLayout.CENTER);

      pack ();
      setVisible (true);
   }

   public static void main (String [] args)
   {
      Runnable r = new Runnable ()
                   {
                       public void run ()
                       {
                          new Notepad ();
                       }
                   };
      EventQueue.invokeLater (r);
   }
}

class NewAction extends AbstractAction
{
   JTextArea document;

   NewAction (JTextArea document)
   {
      this.document = document;

      putValue (NAME, "New");
      putValue (MNEMONIC_KEY, KeyEvent.VK_N);
      putValue (SMALL_ICON, new ImageIcon ("newicon_16x16.gif"));

      // Before Java SE 6, an action's SMALL_ICON key was used to assign the
      // same icon to a button and a menu item. Java SE 6 now makes it
      // possible to assign different icons to these components. If an icon
      // is added via LARGE_ICON_KEY, this icon appears on buttons, whereas
      // an icon added via SMALL_ICON appears on menu items. However, if there
      // is no LARGE_ICON_KEY-based icon, the SMALL_ICON-based icon is
      // assigned to a toolbar's button (for example), in addition to a menu
      // item.
```

```
      putValue (LARGE_ICON_KEY, new ImageIcon ("newicon_32x32.gif"));
   }

   public void actionPerformed (ActionEvent e)
   {
      document.setText ("");
   }
}

class StatAction extends AbstractAction
{
   private JFrame frame;
   private JLabel labelStatus = new JLabel ("Notepad 1.0");

   StatAction (JFrame frame)
   {
      this.frame = frame;

      putValue (NAME, "Status Bar");
      putValue (MNEMONIC_KEY, KeyEvent.VK_A);

      // By default, a mnemonic decoration is presented under the leftmost
      // character in a string having multiple occurrences of this character.
      // For example, the previous putValue (MNEMONIC_KEY, KeyEvent.VK_A);
      // results in the "a" in "Status" being decorated. If you prefer to
      // decorate a different occurrence of a letter (such as the "a" in
      // "Bar"), you can now do this thanks to Java SE 6's
      // displayedMnemonicIndex property and DISPLAYED_MNEMONIC_INDEX_KEY. In
      // the code below, the zero-based index (8) of the "a" appearing in
      // "Bar" is chosen as the occurrence of "a" to receive the decoration.

      putValue (DISPLAYED_MNEMONIC_INDEX_KEY, 8);

      // Java SE 6 now makes it possible to choose the initial selection state
      // of a toggling component. In this application, the component is a
      // JCheckBoxMenuItem that is responsible for determining whether or not
      // to display a status bar. Initially, the status bar will not be shown,
      // which is why false is assigned to the selected property in the method
      // call below.
```

```
        putValue (SELECTED_KEY, false);

        labelStatus = new JLabel ("Notepad 1.0");
        labelStatus.setBorder (new EtchedBorder ());
    }

    public void actionPerformed (ActionEvent e)
    {
        // Because a component updates the selected property, it is easy to find
        // out the current selection setting, and then use this setting to
        // either add or remove the status bar.

        Boolean selection = (Boolean) getValue (SELECTED_KEY);
        if (selection)
            frame.getContentPane ().add (labelStatus, BorderLayout.SOUTH);
        else
            frame.getContentPane ().remove (labelStatus);

        frame.getRootPane ().revalidate ();
    }
}
```

The numerous comments in the source code explain the new action keys and the `setHideActionText()` method. However, you might be curious about my deferring the creation of a Swing application's GUI to the event-dispatching thread, via a `Runnable` instance and the `EventQueue.invokeLater (r);` method call. I do this here (and elsewhere in the book) because creating a Swing GUI on any thread other than the event-dispatching thread—such as an application's main thread or the thread that invokes an applet's `public void init()` method—is unsafe.

■ **Note** Although you could invoke `SwingUtilities.invokeLater()` to ensure that an application's Swing-based GUI is created on the event-dispatching thread, it is somewhat more efficient to invoke `EventQueue.invokeLater()`, because the former method contains a single line of code that calls the latter method. It is also somewhat more efficient to invoke `EventQueue.invokeAndWait()`, rather than `SwingUtilities.invokeAndWait()`, to create an applet's Swing-based GUI on the event-dispatching thread.
