

SCJP Exam for J2SE 5

A Concise and Comprehensive
Study Guide for The Sun Certified
Java Programmer Exam



Paul Sanghera, Ph.D.

Apress®

SCJP Exam for J2SE 5: A Concise and Comprehensive Study Guide for The Sun Certified Java Programmer Exam**Copyright © 2006 by Paul Sanghera, Ph.D.**

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-697-5

ISBN-10 (pbk): 1-59059-697-8

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

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Steve Anglin

Technical Reviewer: Simon Liu

Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick,

Jonathan Hassell, James Huddlestone, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Keir Thomas, Matt Wade

Project Manager: Kylie Johnston

Copy Edit Manager: Nicole LeClerc

Copy Editor: Bill McManus

Assistant Production Director: Kari Brooks-Copony

Production Editor: Laura Cheu

Compositor: Lynn L'Heureux, M&M Composition, LLC

Proofreader: Kim Burton

Indexer: Julie Grady

Artist: April Milne

Cover Designer: Kurt Krames

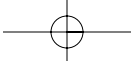
Manufacturing Director: Tom Debolski

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>.

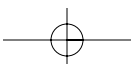
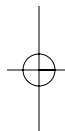
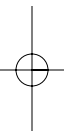
For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

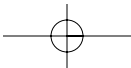
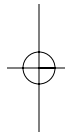
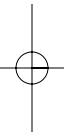
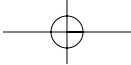
The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

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



*To all the Java enthusiasts across the oceans
To all the programmers of the Java lang
From Mount Everest to Sierra Mountains
From Madagascar to New York Island*





Contents at a Glance

| | |
|------------------------------------|-------|
| About the Author | xvi |
| About the Technical Reviewer | xvii |
| Acknowledgments | xviii |
| Introduction | xix |

PART 1 ■ ■ ■ Scratching the Surface

| | | |
|-------------|--|---|
| ■ CHAPTER 1 | Fundamentals of Java Programming | 3 |
|-------------|--|---|

PART 2 ■ ■ ■ Basic Java Programming

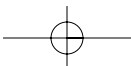
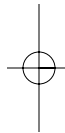
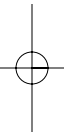
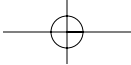
| | | |
|-------------|--|-----|
| ■ CHAPTER 2 | Data Types and Operators | 23 |
| ■ CHAPTER 3 | Classes, Methods, and Interfaces | 57 |
| ■ CHAPTER 4 | Java Language Fundamentals | 89 |
| ■ CHAPTER 5 | Object-Oriented Programming | 121 |
| ■ CHAPTER 6 | Execution Flow Control in Java | 153 |

PART 3 ■ ■ ■ Advanced Java Programming

| | | |
|--------------|---|-----|
| ■ CHAPTER 7 | Exceptions and Assertions | 175 |
| ■ CHAPTER 8 | Input and Output in Java | 197 |
| ■ CHAPTER 9 | Strings, Formatters, and Wrappers | 221 |
| ■ CHAPTER 10 | Collections and Generics | 259 |
| ■ CHAPTER 11 | Threads in Java | 291 |

PART 4 ■ ■ ■ Appendixes

| | | |
|---------------|---|-----|
| ■ APPENDIX A | Installing and Testing J2SE 5.0 | 317 |
| ■ APPENDIX B | SCJP 5.0 Upgrade Exam (CX-310-056) | 321 |
| ■ APPENDIX C | Answers to Chapter Review Questions | 325 |
| ■ APPENDIX D | Practice Exam | 333 |
| ■ APPENDIX E | Exam Quick Prep | 363 |
| ■ INDEX | | 385 |



Contents

| | |
|------------------------------------|-------|
| About the Author | xvi |
| About the Technical Reviewer | xvii |
| Acknowledgments | xviii |
| Introduction | xix |

PART 1 ■ ■ ■ Scratching the Surface

| | | |
|-------------|---|----|
| ■ CHAPTER 1 | Fundamentals of Java Programming | 3 |
| | Learning Objectives | 3 |
| | Computers and Computer Programming | 4 |
| | How a Computer Works | 4 |
| | How a Computer Program Works | 5 |
| | Writing and Executing a Java Program | 6 |
| | Writing a Java Program | 6 |
| | Compiling a Java Program | 7 |
| | Executing a Java Program | 7 |
| | Elements of a Java Program | 9 |
| | Classes and Objects | 9 |
| | Methods | 10 |
| | Variables and Data Types | 11 |
| | Execution Flow of a Program | 12 |
| | Features of Java | 13 |
| | Platform Independence | 13 |
| | Object-Oriented Programming | 14 |
| | Accessing the Classes and Class Members | 15 |
| | The Memory Usage by a Java Program | 16 |
| | Living on the Stack | 17 |
| | Living on the Heap | 17 |
| | When Will the Error Occur? | 17 |
| | What's in a Name? | 18 |
| | A Brief History of Java Versions | 18 |
| | Java Buzzwords | 19 |
| | Summary | 20 |

PART 2 ■ ■ ■ Basic Java Programming

| | | |
|------------------|--|----|
| CHAPTER 2 | Data Types and Operators | 23 |
| | Exam Objectives | 23 |
| | Data-Related Concepts | 23 |
| | Understanding Variables, Data Types, and Operators | 23 |
| | Naming the Variables: Legal Identifiers | 24 |
| | Reserved Names: The Keywords | 25 |
| | Working with Primitive Data Types | 26 |
| | Declaring and Initializing Primitive Variables | 27 |
| | Syntax for Declaring Variables | 27 |
| | Accessing Variables | 27 |
| | Literals | 28 |
| | Default Initial Values | 30 |
| | Working with Nonprimitive Data Types | 31 |
| | Objects | 31 |
| | Arrays | 32 |
| | The Data Type enum | 33 |
| | Understanding Operations on Data | 36 |
| | Arithmetic Operators | 36 |
| | The Unary Arithmetic Operators | 37 |
| | Basic Arithmetic Operators | 38 |
| | Relational Operators | 40 |
| | Logical Operators | 41 |
| | Bitwise Logical Operators | 41 |
| | Short-Circuit Logical Operators | 43 |
| | Using Assignment Operators | 45 |
| | Shortcut Assignment Operators | 45 |
| | Arithmetic Promotion | 46 |
| | Advanced Operators | 47 |
| | The Shortcut if-else Operator: ?: | 47 |
| | The Cast Operator: (<type>) | 47 |
| | The new Operator | 48 |
| | The instanceof Operator | 48 |
| | Equality of Two Objects or Two Primitives | 49 |
| | Codewalk Quicklet | 49 |
| | Summary | 50 |
| | Exam's Eye View | 51 |
| | Review Questions | 52 |

| | | |
|------------------|---|-----|
| CHAPTER 3 | Classes, Methods, and Interfaces | 57 |
| | Exam Objectives | 57 |
| | Using Methods | 58 |
| | Defining a Method | 58 |
| | The Static Methods and Variables | 59 |
| | Methods with a Variable Number of Parameters | 61 |
| | JavaBeans Naming Standard for Methods | 63 |
| | Working with Classes and Objects | 64 |
| | Defining Classes | 64 |
| | Creating Objects | 65 |
| | Nested Classes | 67 |
| | Understanding Enums | 69 |
| | Methods of the Enum Class | 70 |
| | Constructors, Methods, and Variables in an enum | 70 |
| | Inheritance | 73 |
| | Writing and Invoking Constructors | 75 |
| | Writing and Using Interfaces | 78 |
| | Codewalk Quicklet | 81 |
| | Summary | 82 |
| | Exam's Eye View | 83 |
| | Review Questions | 84 |
| CHAPTER 4 | Java Language Fundamentals | 89 |
| | Exam Objectives | 89 |
| | Organizing Your Java Application | 90 |
| | Entering Through the Main Gate | 90 |
| | What Is in a Name? | 92 |
| | The JAR Files | 95 |
| | The Static Import | 96 |
| | Passing Arguments into Methods | 97 |
| | Passing a Primitive Variable | 97 |
| | Passing a Reference Variable | 98 |
| | Using Access Modifiers | 99 |
| | The public Modifier | 100 |
| | The private Modifier | 100 |
| | The protected Modifier | 101 |
| | The Default Modifier | 102 |

| | |
|---|------------|
| Understanding Usage Modifiers | 104 |
| The final Modifier | 104 |
| The static Modifier | 105 |
| The abstract Modifier | 107 |
| The native Modifier | 108 |
| The transient Modifier | 109 |
| The Thread-Related Modifiers | 110 |
| Modifiers: The Big Picture | 110 |
| Understanding Garbage Collection in Java | 111 |
| Understanding the Garbage Collector | 111 |
| The finalize() Method | 112 |
| Codewalk Quicklet | 113 |
| Summary | 114 |
| Exam's Eye View | 115 |
| Review Questions | 116 |
| | |
| CHAPTER 5 | |
| Object-Oriented Programming | 121 |
| Exam Objectives | 121 |
| Understanding Object-Oriented Relationships | 122 |
| The is-a Relationship | 122 |
| The has-a Relationship | 122 |
| Encapsulation and Data Abstraction | 123 |
| Coupling and Cohesion | 124 |
| Implementing Polymorphism | 125 |
| Conversion of Primitive Data Types | 127 |
| Implicit Conversion of Primitive Data Types | 127 |
| Explicit Conversion of Primitive Data Types | 133 |
| Conversion of Object Reference Types | 134 |
| Implicit Conversion of Object Reference Types | 134 |
| Explicit Conversion of Object Reference Types | 137 |
| Using Method Overriding and Overloading | 138 |
| Method Overriding | 138 |
| Method Overloading | 140 |
| Constructor Overloading | 143 |
| Codewalk Quicklet | 144 |
| Summary | 145 |
| Exam's Eye View | 146 |
| Review Questions | 147 |

| | | |
|------------------|---|-----|
| CHAPTER 6 | Execution Flow Control in Java | 153 |
| | Exam Objectives | 153 |
| | Using Selection Statements | 153 |
| | The if Statements | 153 |
| | The switch Statement | 157 |
| | Iteration Statements | 160 |
| | The while Loop Construct | 160 |
| | The do-while Loop Construct | 161 |
| | The for Loop Construct | 161 |
| | The for-each Loop Construct | 163 |
| | Block Breaker Statements | 164 |
| | The continue Statement | 164 |
| | The break Statement | 165 |
| | Codewalk Quicklet | 167 |
| | Summary | 167 |
| | Exam's Eye View | 168 |
| | Review Questions | 169 |

PART 3 ■ ■ ■ Advanced Java Programming

| | | |
|------------------|---|-----|
| CHAPTER 7 | Exceptions and Assertions | 175 |
| | Exam Objectives | 175 |
| | Understanding Exceptions in Java | 175 |
| | The Exception Tree in Java | 176 |
| | Checked Exceptions and Runtime Exceptions | 177 |
| | Standard Exceptions | 178 |
| | Basics of Exception Handling | 178 |
| | Using the try and catch Blocks | 179 |
| | Using the finally Block | 180 |
| | Using Multiple catch Blocks | 181 |
| | Throwing Exceptions | 183 |
| | Control Flow in Exception Condition | 184 |
| | Declaring Exceptions | 186 |
| | Checked Exception: Duck It or Catch It | 186 |
| | Declaring Exceptions When Overriding | 187 |
| | Assertions | 188 |
| | Codewalk Quicklet | 190 |
| | Summary | 191 |
| | Exam's Eye View | 192 |
| | Review Questions | 193 |

| | | |
|------------------|--|-----|
| CHAPTER 8 | Input and Output in Java | 197 |
| | Exam Objectives | 197 |
| | Handling Files in Java | 197 |
| | Understanding the File Class | 198 |
| | Navigating the File System | 200 |
| | Understanding Streams | 202 |
| | The Low-Level Streams | 202 |
| | The High-Level Streams | 205 |
| | Readers and Writers | 208 |
| | Low-Level Readers and Writers | 209 |
| | High-Level Readers and Writers | 211 |
| | Exceptions During I/O Operations | 213 |
| | Object Streams and Serialization | 213 |
| | Writing with ObjectOutputStream | 214 |
| | Reading with ObjectInputStream | 214 |
| | Codewalk Quicklet | 215 |
| | Summary | 216 |
| | Exam's Eye View | 217 |
| | Review Questions | 218 |
| | | |
| CHAPTER 9 | Strings, Formatters, and Wrappers | 221 |
| | Exam Objectives | 221 |
| | Using the String Class | 222 |
| | Constructing Strings with the String Class | 222 |
| | Methods of the String Class | 223 |
| | The Immutability of Strings | 225 |
| | The StringBuffer Class | 227 |
| | Formatting and Parsing for the World | 229 |
| | Definitions of Internationalization and Localization | 229 |
| | Understanding the Locale Class | 229 |
| | Formatting Numbers and Currencies | 232 |
| | Formatting Dates | 235 |
| | Text Processing with Regular Expressions | 236 |
| | Formatting and Parsing Streams | 240 |
| | Formatting Streams | 240 |
| | Parsing Streams | 244 |
| | Wrapping the Primitives | 246 |
| | Creating Objects of Wrapper Classes | 246 |
| | Methods to Extract the Wrapped Values | 248 |
| | The Instant Use of Wrapper Classes | 249 |

| | |
|--|------------|
| Codewalk Quicklet | 251 |
| Summary | 252 |
| Exam's Eye View | 253 |
| Review Questions | 254 |
| | |
| CHAPTER 10 Collections and Generics | 259 |
| Exam Objectives | 259 |
| The Mother of All Classes: Object | 260 |
| The Object Class | 260 |
| The equals() Method | 261 |
| The toString() Method | 263 |
| Understanding Collections | 263 |
| The Collections Interfaces | 264 |
| Implementations of Collections Interfaces | 266 |
| The hashCode Method | 270 |
| Understanding Generics | 274 |
| Generic Collections | 274 |
| Generic Programming | 276 |
| Object Ordering | 279 |
| Natural Ordering | 279 |
| Defining Ordering Using Comparator | 280 |
| Understanding Autoboxing | 281 |
| Codewalk Quicklet | 283 |
| Summary | 285 |
| Exam's Eye View | 286 |
| Review Questions | 287 |
| | |
| CHAPTER 11 Threads in Java | 291 |
| Exam Objectives | 291 |
| Multithreaded Programming in Java | 291 |
| Understanding Threads | 291 |
| Creating a Thread Using the Thread Class | 292 |
| Creating a Thread Using the Runnable Interface | 294 |
| Spawning Multiple Threads | 296 |
| Lifecycle of a Thread: An Overview | 297 |
| Understanding Transition Between Thread States | 298 |
| Transition Between Running and Runnable States | 299 |
| Transition Between Runnable and Nonrunnable States | 299 |

| | |
|--|-----|
| Synchronization and Locks in Concurrent Access | 301 |
| Understanding the Concurrent Access Problem | 301 |
| Object Locks | 302 |
| Class Locks | 303 |
| Monitoring the Wait State | 304 |
| Scheduling Threads | 307 |
| Understanding the Deadlocks | 308 |
| Codewalk Quicklet | 309 |
| Summary | 310 |
| Exam's Eye View | 311 |
| Review Questions | 312 |

PART 4 ■ ■ ■ **Appendixes**

| | | |
|---------------------|--|-----|
| ■ APPENDIX A | Installing and Testing J2SE 5.0 | 317 |
| | Installing the Required Java Software | 317 |
| | Setting Up the Development Environment Variables | 318 |
| | Writing, Compiling, and Executing a Java Program | 318 |
| ■ APPENDIX B | SCJP 5.0 Upgrade Exam (CX-310-056) | 321 |
| | Essential Information About the Upgrade Exam | 321 |
| | Comparison Between the Regular and Upgrade Exams | 322 |
| | Upgrade Exam Objectives: Where Are They Covered? | 322 |
| ■ APPENDIX C | Answers to Chapter Review Questions | 325 |
| | Chapter 2: Data Types and Operators | 325 |
| | Chapter 3: Classes, Methods, and Interfaces | 326 |
| | Chapter 4: Java Language Fundamentals | 327 |
| | Chapter 5: Object-Oriented Programming | 327 |
| | Chapter 6: Execution Flow Control in Java | 328 |
| | Chapter 7: Exceptions and Assertions | 329 |
| | Chapter 8: Input and Output in Java | 330 |
| | Chapter 9: Strings, Formatters, and Wrappers | 330 |
| | Chapter 10: Collections and Generics | 331 |
| | Chapter 11: Threads in Java | 332 |

| | | |
|--------------|---|-----|
| ■ APPENDIX D | Practice Exam | 333 |
| | Questions | 333 |
| | Answers and Explanations | 358 |
| ■ APPENDIX E | Exam Quick Prep | 363 |
| | Chapter 2: Data Types and Operators (Exam Objectives 1.3, 7.6) | 363 |
| | Chapter 3: Classes, Methods, and Interfaces (Exam Objectives 1.1, 1.2, 1.4, 1.6) | 364 |
| | Chapter 4: Classes, Methods, and Interfaces (Exam Objectives 5.3, 7.1–7.5) | 367 |
| | Chapter 5: Object-Oriented Programming (Exam Objectives 1.5, 5.1, 5.2, 5.4, 5.5) | 369 |
| | Chapter 6: Execution Flow Control in Java (Exam Objectives 2.1, 2.2) | 371 |
| | Chapter 7: Exceptions and Assertions (Exam Objectives 2.3–2.6) | 373 |
| | Chapter 8: Input and Output in Java (Exam Objectives 3.2, 3.3) | 375 |
| | Chapter 9: Strings, Formatters, and Wrappers (Exam Objectives 3.1, 3.4, 3.5) | 376 |
| | Chapter 10: Collections and Generics (Exam Objectives 6.1–6.5) | 378 |
| | Chapter 11: Threads in Java (Exam Objectives 4.1–4.4) | 381 |
| ■ INDEX | | 385 |

About the Author



■ **PAUL SANGHERA**, Ph.D., SCJP, SCBCD, who contributed to developing the SCJP exam for Java 5, has been programming in Java for 10 years and has substantial experience teaching Java. As a software engineer, Paul has contributed to the development of world-class technologies such as Netscape Communicator and Novell's NDS. He has been director of software development and director of project management at successful startups such as WebOrder and MP3.com. He has taught Java and other technology courses at several institutes in the San Francisco Bay Area, including San Jose State University, Golden Gate University, California State University, Hayward, and Brooks College. With a master's degree in computer science from Cornell University and a Ph.D. in physics from Carleton University, he has authored and co-authored more than 100 technical papers published in well-reputed European and American research journals. Paul has also presented talks by invitation at several international scientific conferences. He is the best-selling author of several books on technology and project management. Paul lives in Silicon Valley, California, where he works as an independent information consultant.

About the Technical Reviewer

■ **SIMON LIU** has worked with Java for six years and mainly has developed Java-based financial applications. He loves to read technical books and has reviewed several certification books.

Simon received bachelor's and master's degrees in Computer Science from the University of Hong Kong, and has acquired several certificates, including SCJP, SCJA, SCWCD, SCBCD, SCDJWS, SCMAD, ICSD, ICED, ICDBA, and OCP.

Acknowledgments

As they say (well, if they don't any more, they should), first things first. Let me begin by thanking Steve Anglin, whose e-mail message triggered this project. With two thumbs up, thanks to Kylie Johnston, the project manager of this book, for her focus, dedication, professionalism, and results-oriented approach.

It takes a team to materialize a book idea into a published book. It is my great pleasure to acknowledge the hard and smart work of the Apress team that made it happen. Here are a few names to mention: Bill McManus for copy editing, Laura Cheu for managing the production process, Lynn L'Heureux for compositing, Kim Burton for proofreading, and Julie Grady for indexing. My special thanks to Stephanie Parker, the marketing manager for this book, for bridging the gap between the author and the reader. The actions of all these folks spoke to me in one voice: Apress means Author's Press. I am thankful to Simon Liu, the technical editor of this book, for doing an excellent job in thoroughly reviewing the manuscript and offering valuable feedback.

In some ways, writing this book is an expression of the technologist and educator inside me. I thank my fellow technologists who guided me at various places during my journey in the computer industry from Novell to Dream Logic: Chuck Castleton at Novell, Delon Dotson at Netscape and MP3.com, Kate Peterson at WebOrder, and Dr. John Serri at Dream Logic. I also thank my colleagues and seniors in the field of education for helping me in so many ways to become a better educator. Here are a few to mention: Dr. Gerald Pauler (Brooks College), Professor David Hayes (San Jose State University), Professor Michael Burke (San Jose State University), and Dr. John Serri (University of Phoenix).

Friends always lend a helping hand, in many visible and invisible ways, in almost anything important we do in our lives. Without them, the world would be a very boring and uncreative place. Here are a few I would like to mention: Stanley Wong, Patrick Smith, Kulwinder, Major Bhupinder Singh Daler, Ruth Gordon, Srilatha Moturi, Baldev Khullar, and the Kandola family (Gurmail and Sukhwinder).

Last, but not least, my appreciation (along with my heart) goes to my wife Renee and my son Adam for not only peacefully coexisting with my book projects but also supporting them.

Introduction

I have made this letter longer than usual, only because I have not had the time to make it shorter.

Blaise Pascal

This book covers the topics determined by the exam objectives for the Sun Certified Java Programmer (SCJP) for Java 5 exam, CX-310-055. Each chapter explores topics in Java programming specified by a set of exam objectives in a manner that makes the presentation cohesive, concise, and yet comprehensive.

Who This Book Is For

This book is primarily targeted at the Java programmers and students who want to prepare for the SCJP certification exam for Java 5, CX-310-055, or the update exam, CX-310-056. Since the book has a laser-sharp focus on the exam objectives, expert Java programmers who want to pass the exam can use this book to ensure that they do not overlook any objective. Yet, it is not an exam-cram book. The chapters and the sections inside each chapter are presented in a logical learning sequence: every new chapter builds upon knowledge acquired in previous chapters, and there is no hopping from topic to topic. The concepts and topics, simple and complex, are explained in a concise yet comprehensive fashion. This facilitates stepwise learning and prevents confusion. Furthermore, Chapter 1 presents a very basic introduction to computer programming and the Java programming language for absolute beginners. Hence, this book is also very useful for beginners to get up to speed quickly even if they are new to Java and computer programming. Even after the exam, you will find yourself returning to this book as a useful reference for basic Java programming.

In a nutshell, this book can be used by the following audiences:

- Beginners with no prior Java experience can use this book to learn basic Java programming, pass the SCJP exam, or both.
- Advanced Java programmers who want to pass the SCJP exam can use this book to ensure they don't miss any exam objectives.
- Instructors teaching a first course in Java can use this book as a text book.

How This Book Is Structured

The structure of this book is determined by the following two requirements:

- The book is equally useful for both beginners and experts who want to pass the SCJP exam for Java 5.
- Although it has a laser-sharp focus on the exam objectives, the book is not an exam cram. It presents the material in a logical learning sequence so that the book can be used for learning (or teaching) basic Java programming.

This book has four parts:

| Part | Topic | Chapters/Appendixes |
|------|--|---------------------|
| 1 | Introduction to computer programming and Java | 1 |
| 2 | Basic Java programming | 2 through 6 |
| 3 | Advanced topics in Java programming | 7 through 11 |
| 4 | Appendixes, including a complete practice exam, answers to review questions, and Exam Quick Prep | A, B, C, D, and E |

How Each Chapter Is Organized

With the exception of Chapter 1, which covers the basics of computer programming and Java, each chapter begins with a list of exam objectives on which the chapter is focused. I have somewhat rearranged the order of the objectives to keep the topics and the subject matter in line with sequential learning and to avoid hopping from topic to topic.

Each chapter starts with an introduction that establishes the concepts or topics that will be explored in the chapter. As you read through a chapter, you will find the following features:

- *Notes*: Emphasize important concepts or information.
- *Cautions*: Point out information that may be contrary to your expectations depending upon your level of experience with Java programming. Both Notes and Cautions are important from the exam viewpoint.
- *Summary*: This section provides the big picture and reviews the important concepts in the chapter.
- *Exam's-Eye View*: This section highlights the important points in the chapter from the perspective of the exam: the information that you must comprehend, the things that you should look out for because they might seem counterintuitive, and the facts that you should memorize for the exam.
- *Review Questions*: This section has a two-pronged purpose: to help you test your knowledge about the material presented in the chapter, and to help you evaluate your ability to answer the exam questions based on the exam objectives covered in the chapter. The answers to the review questions are presented in Appendix C.

A single feature of the SCJP exam that makes it difficult is that it is very code intensive. In order to succeed in the exam, you must develop stamina for reading and understanding code. To raise your comfort level with the code, this book offers the following three unique features:

- *Complete code examples*: Most of the code examples in the book are complete, tested, and runnable programs that you can download and experiment with.
- *Code for the practice exam*: The code for the practice exam questions is also provided for download so that you can execute and experiment with it.
- *Codewalk Quicklets*: Each chapter offers a “Codewalk Quicklet” section in which you are encouraged to follow a process-based codewalk, which is a way of looking at the code from the perspective of a process. A process, by definition, has an input, operation on the input, and output as a result of the operation. The focus here is not necessarily the complexity of the code, but rather a way of looking at the code. If you develop this way of looking at the code, you will be able to answer most of the code-intensive questions on the exam in an efficient and effective manner.

Other special features of the book are the following:

- A complete practice exam (Appendix D) with questions modeled after the real exam and fully explained answers
- An Exam Quick Prep (Appendix E) that recaps all the important points for the last hour of preparation before taking the exam
- An appendix (Appendix B) that provides useful information and analysis for programmers who are considering updating their J2SE 1.4 certification to J2SE 5

This book and the exam are based on Java 2 Standard Edition (J2SE) 5.0, which you can download from the Sun website, install on your computer, and test the installation as described in Appendix A. You will be using this environment to try the code examples in this book, and in the practice exam.

Conventions

The following are some of the conventions used in this book:

- Conventions used in referring to methods are as follows:
 - When a method name ends with (...), it means the method has one or more arguments.
 - When a method name ends with (), it means the method may or may not have one or more arguments.
- In presenting the syntax, a word in angle brackets (<>) represents a variable part of a construct. You must provide its value when actually using it in your program. The generic programming explained in Chapter 10 is an exception to this convention, and has its own meaning for the angle brackets.

Downloading the Code

The SCJP exam is very code intensive. To pass the exam, it's absolutely imperative that you feel comfortable with the code under time pressure. To help you with that, this book offers complete runnable programs corresponding to the examples in the book and the questions in the practice exam. You are recommended to actually execute these programs and experiment with them to find the answers to the questions that may pop up as you are preparing for the exam.

The following downloads are available:

- Source code for the programming examples in the book chapters
- Source code for the programming examples in the practice exam

You can download these items from the Source Code area of the Apress website (www.apress.com).

About the Exam

With the popularity of Java in the enterprise, SCJP certification is an important credential for a Java programmer to earn. It is also a prerequisite to the whole spectrum of specialty certifications in Java available from Sun.

The Java Certification Exams from Sun

The Java platform comes in three flavors: Java 2 Standard Edition (J2SE), Java 2 Enterprise Edition (J2EE), and Java 2 Micro Edition (J2ME). As shown in Figure 1, the certification paths based on these platforms include exams for the following certifications: Sun Certified Java Programmer (SCJP), Sun Certified Java Developer (SCJD), Sun Certified Web Component Developer (SCWCD), Sun Certified Business Component Developer (SCBCD), Sun Certified Developer for Java Web Services (SCDJWS), and Sun Certified Mobile Application Developer (SCMAD). The SCJP is the prerequisite for all other certifications in this list.

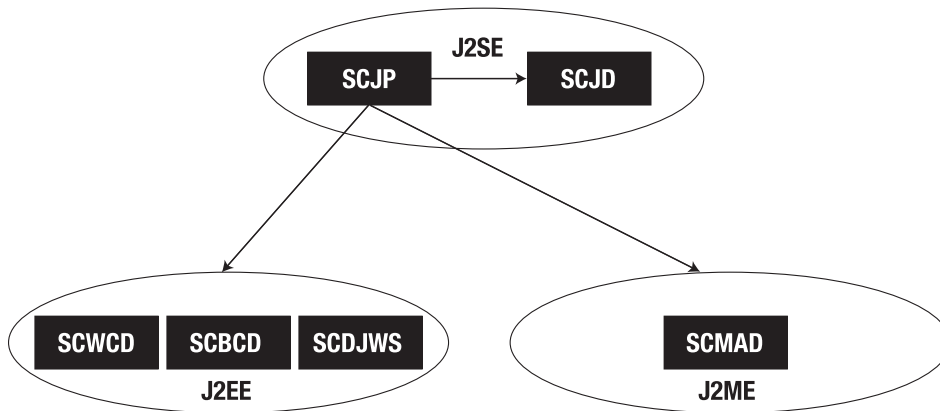


Figure 1. Certification paths based on the J2SE, J2EE, and J2ME platforms. SCJP certification is a prerequisite for all other certifications shown here.

The SCJP exam is not a prerequisite only for the following two Java certifications:

- Sun Certified Java Associate (SCJA)
- Sun Certified Enterprise Architect for J2EE Technology (SCEA)

Preparing for the SCJP Exam

The SCJP for Java 5 certification is the first Java certification on the career path in Java. Achieving this certification provides evidence that you understand the basic syntax and structure of the Java programming language and that you can create Java applications using J2SE 5.0 to run on server and desktop systems.

This exam is for programmers who have experience using the Java programming language. If you are a beginner, you will learn Java programming while preparing for the exam because this book is not a mere exam cram. On the other end of the spectrum, even an expert Java programmer may fail this exam if not prepared for it properly. From the exam point of view, pay special attention to the following items while preparing for the exam:

- Carefully read the exam objectives at the beginning of each chapter.
- Make sure you understand the Notes and Cautions in each chapter.
- Study the review questions at the end of each chapter.
- Take the practice exam in Appendix D toward the end your exam preparation.
- Review the “Exam’s-Eye View” sections and the Exam Quick Prep in Appendix E during the last hours of your preparation.

The most important point to remember for this exam is this: it is a code-intensive exam. Make sure your comfort level with the code is such that if you encounter a code fragment of up to 30 lines, you can read it and understand it without panic.

Taking the SCJP Exam

The SCJP certification consists of one exam available at authorized Prometric Testing Centers throughout the world. Following are some important details about the exam:

- Exam ID: CX-310-055
- Prerequisite: None
- Cost: \$150 (which may vary by country and if you have discount coupons)
- Number of questions: 72
- Pass score: 59 percent; that is, answer at least 43 questions correctly out of 72
- Maximum time allowed: 175 minutes

The question format is multiple choice, including some drag-and-drop questions. In most of the questions, you are asked to select the correct answer or answers from multiple answer choices presented for a question. The number of correct answers is given. Pay attention to the Exhibit button if it appears in a question. You click it to get the required information for the question. Drag and drop is a special kind of multiple-choice question that requires you to choose the right box from multiple boxes and drag it to the right spot.

According to many exam takers, there are hardly any questions with true/false answers, a lot of questions with three (or sometimes two) correct answers, and quite a few questions with radio button answers. A question with radio button answers means that only one answer is correct. The questions vary from very easy to very difficult, but mostly difficult.

If you can't make up your mind about the answer to a question, you can skip it and come back to it later. That way, you avoid running out of time while there are still some easy questions waiting for you. Make sure you understand when a compilation error is generated and when an exception is thrown at runtime. You will encounter both of these choices in the answer options to most of the questions.

Above all, make sure you feel comfortable reading and understanding code. The code in the exam will mostly have the code lines numbered. If the numbering starts from 1, it means the entire code from a source file is there. When the first code line is numbered with a number greater than 1, that means you are looking at only a part of the code from the source file. In this case, you can assume that the code you cannot see is correct.

Like other Sun exams, the SCJP exam starts with a survey that asks you questions regarding your level of knowledge and experience with different topics on Java technology. Please don't get confused or panic; rest assured that this is not part of the actual exam.

Following are the main steps in the process of taking the exam:

1. You should purchase an exam voucher from your local Sun Education Services Office. You can also purchase the voucher online by going to <http://suned.sun.com/US/certification/register/index.html>.
2. The exams are conducted by Prometric all across the world, and you need to schedule your exam time and location with them. After you have purchased the exam voucher, contact an authorized Prometric Testing Center near you. You can get the information from www.prometric.com.
3. Reach the testing center at least 15 minutes before the test start time, and be prepared to show two forms of identification, one of which should be a photo ID.

4. After you finish the test, the computer screen will display your result (whether you have passed or not). You will also receive a printed copy of the detailed results.
5. Within a month or so, you will receive your certificate from Sun in the mail if you passed the exam.

For current and complete information on the exam, you can visit the Sun exam site: www.sun.com/training/certification/.
Best wishes for the exam. Go for it!

Contacting the Author

More information about Dr. Paul Sanghera can be found at www.paulsanghera.com. He can be reached at: paul_s_sanghera@yahoo.com.

Exam Readiness Checklist: Exam CX-310-055

| Exam Objective | Chapter Number |
|---|----------------|
| <p>1.1 Develop code that declares classes (including abstract and all forms of nested classes), interfaces, and enums, and includes the appropriate use of package and import statements (including static imports).</p> <p>1.2 Develop code that declares an interface. Develop code that implements or extends one or more interfaces. Develop code that declares an abstract class. Develop code that extends an abstract class.</p> | 3, 4 |
| <p>1.3 Develop code that declares, initializes, and uses primitives, arrays, enums, and objects as static, instance, and local variables. Also, use legal identifiers for variable names.</p> | 2 |
| <p>1.4 Develop code that declares both static and non-static methods, and—if appropriate—use method names that adhere to the JavaBeans naming standards. Also develop code that declares and uses a variable-length argument list.</p> | 3 |
| <p>1.5 Given a code example, determine if a method is correctly overriding or overloading another method, and identify legal return values (including covariant returns), for the method.</p> | 5 |
| <p>1.6 Given a set of classes and superclasses, develop constructors for one or more of the classes. Given a class declaration, determine if a default constructor will be created, and if so, determine the behavior of that constructor. Given a nested or non-nested class listing, write code to instantiate the class.</p> | 3 |
| <p>2.1 Develop code that implements an if or switch statement; and identify legal argument types for these statements.</p> <p>2.2 Develop code that implements all forms of loops and iterators, including the use of for, the enhanced for loop (for-each), do, while, labels, break, and continue; and explain the values taken by loop counter variables during and after loop execution.</p> | 6 |

| Exam Objective | Chapter Number |
|---|-----------------------|
| 2.3 Develop code that makes use of assertions, and distinguish appropriate from inappropriate uses of assertions. | 7 |
| 2.4 Develop code that makes use of exceptions and exception handling clauses (try, catch, finally), and declares methods and overriding methods that throw exceptions. | |
| 2.5 Recognize the effect of an exception arising at a specified point in a code fragment. Note that the exception may be a runtime exception, a checked exception, or an error. | |
| 2.6 Recognize situations that will result in any of the following being thrown: <code>ArrayIndexOutOfBoundsException</code> , <code>ClassCastException</code> , <code>IllegalArgumentException</code> , <code>IllegalStateException</code> , <code>NullPointerException</code> , <code>NumberFormatException</code> , <code>AssertionError</code> , <code>ExceptionInInitializerError</code> , <code>StackOverflowError</code> , or <code>NoClassDefFoundError</code> . Understand which of these are thrown by the virtual machine and recognize situations in which others should be thrown programmatically. | |
| 3.1 Develop code that uses the primitive wrapper classes (such as <code>Boolean</code> , <code>Character</code> , <code>Double</code> , <code>Integer</code> , etc.), and/or autoboxing & unboxing. Discuss the differences between the <code>String</code> , <code>StringBuilder</code> , and <code>StringBuffer</code> classes. | 9 |
| 3.2 Given a scenario involving navigating file systems, reading from files, or writing to files, develop the correct solution using the following classes (sometimes in combination), from <code>java.io</code> : <code>BufferedReader</code> , <code>BufferedWriter</code> , <code>File</code> , <code>FileReader</code> , <code>FileWriter</code> , and <code>PrintWriter</code> . | 8 |
| 3.3 Develop code that serializes and/or de-serializes objects using the following APIs from <code>java.io</code> : <code>DataInputStream</code> , <code>DataOutputStream</code> , <code>FileInputStream</code> , <code>FileOutputStream</code> , <code>ObjectInputStream</code> , <code>ObjectOutputStream</code> , and <code>Serializable</code> . | |
| 3.4 Use standard J2SE APIs in the <code>java.text</code> package to correctly format or parse dates, numbers, and currency values for a specific locale; and, given a scenario, determine the appropriate methods to use if you want to use the default locale or a specific locale. Describe the purpose and use of the <code>java.util.Locale</code> class. | 9 |
| 3.5 Write code that uses standard J2SE APIs in the <code>java.util</code> and <code>java.util.regex</code> packages to format or parse strings or streams. For strings, write code that uses the <code>Pattern</code> and <code>Matcher</code> classes and the <code>String.split</code> method. Recognize and use regular expression patterns for matching (limited to: <code>.</code> (dot), <code>*</code> (star), <code>+</code> (plus), <code>?</code> , <code>\d</code> , <code>\s</code> , <code>\w</code> , <code>[]</code> , <code>()</code>). The use of <code>*</code> , <code>+</code> , and <code>?</code> will be limited to greedy quantifiers, and the parenthesis operator will only be used as a grouping mechanism, not for capturing content during matching. For streams, write code using the <code>Formatter</code> and <code>Scanner</code> classes and the <code>PrintWriter.format/printf</code> methods. Recognize and use formatting parameters (limited to: <code>%b</code> , <code>%c</code> , <code>%d</code> , <code>%f</code> , <code>%s</code>) in format strings. | |
| 4.1 Write code to define, instantiate, and start new threads using both <code>java.lang.Thread</code> and <code>java.lang.Runnable</code> . | 11 |
| 4.2 Recognize the states in which a thread can exist, and identify ways in which a thread can transition from one state to another. | |
| 4.3 Given a scenario, write code that makes appropriate use of object locking to protect static or instance variables from concurrent access problems. | |
| 4.4 Given a scenario, write code that makes appropriate use of <code>wait</code> , <code>notify</code> , or <code>notifyAll</code> . | |
| 5.1 Develop code that implements tight encapsulation, loose coupling, and high cohesion in classes, and describe the benefits. | 5 |
| 5.2 Given a scenario, develop code that demonstrates the use of polymorphism. Further, determine when casting will be necessary and recognize compiler vs. runtime errors related to object reference casting. | |

continued

| Exam Objective | Chapter Number |
|---|-----------------------|
| 5.3 Explain the effect of modifiers on inheritance with respect to constructors, instance or static variables, and instance or static methods. | 4 |
| 5.4 Given a scenario, develop code that declares and/or invokes overridden or overloaded methods and code that declares and/or invokes superclass, overridden, or overloaded constructors. | 5 |
| 5.5 Develop code that implements “is-a” and/or “has-a” relationships. | |
| 6.1 Given a design scenario, determine which collection classes and/or interfaces should be used to properly implement that design, including the use of the Comparable interface. | 10 |
| 6.2 Distinguish between correct and incorrect overrides of corresponding hashCode and equals methods, and explain the difference between == and the equals method. | |
| 6.3 Write code that uses the generic versions of the Collections API, in particular, the Set, List, and Map interfaces and implementation classes. Recognize the limitations of the non-generic Collections API and how to refactor code to use the generic versions. | |
| 6.4 Develop code that makes proper use of type parameters in class/interface declarations, instance variables, method arguments, and return types; and write generic methods or methods that make use of wildcard types and understand the similarities and differences between these two approaches. | |
| 6.5 Use capabilities in the java.util package to write code to manipulate a list by sorting, performing a binary search, or converting the list to an array. Use capabilities in the java.util package to write code to manipulate an array by sorting, performing a binary search, or converting the array to a list. Use the java.util.Comparable and java.lang.Comparable interfaces to affect the sorting of lists and arrays. Furthermore, recognize the effect of the “natural ordering” of primitive wrapper classes and java.lang.String on sorting. | |
| 7.1 Given a code example and a scenario, write code that uses the appropriate access modifiers, package declarations, and import statements to interact with (through access or inheritance) the code in the example. | 4 |
| 7.2 Given an example of a class and a command-line, determine the expected runtime behavior. | |
| 7.3 Determine the effect upon object references and primitive values when they are passed into methods that perform assignments or other modifying operations on the parameters. | |
| 7.4 Given a code example, recognize the point at which an object becomes eligible for garbage collection, and determine what is and is not guaranteed by the garbage collection system. Recognize the behaviors of System.gc and finalization. | |
| 7.5 Given the fully-qualified name of a class that is deployed inside and/or outside a JAR file, construct the appropriate directory structure for that class. Given a code example and a classpath, determine whether the classpath will allow the code to compile successfully. | |
| 7.6 Write code that correctly applies the appropriate operators including assignment operators (limited to: =, +=, -=), arithmetic operators (limited to: +, -, *, /, %, ++, --), relational operators (limited to: <, <=, >, >=, ==, !=), the instanceof operator, logical operators (limited to: &, , ^, !, &&,), and the conditional operator (? :), to produce a desired result. Write code that determines the equality of two objects or two primitives. | 2 |