

Java Concurrency in Practice

By Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes,
Doug Lea

.....
Publisher: **Addison Wesley Professional**
Pub Date: **May 09, 2006**
Print ISBN-10: **0-321-34960-1**
Print ISBN-13: **978-0-321-34960-6**
Pages: **384**

[Table of Contents](#) | [Index](#)

[Copyright](#)

[Advance Praise for Java Concurrency in Practice](#)

[Listings](#)

[Preface](#)

[Chapter 1. Introduction](#)

[Section 1.1. A \(Very\) Brief History of Concurrency](#)

[Section 1.2. Benefits of Threads](#)

[Section 1.3. Risks of Threads](#)

[Section 1.4. Threads are Everywhere](#)

[Part I: Fundamentals](#)

[Chapter 2. Thread Safety](#)

[Section 2.1. What is Thread Safety?](#)

[Section 2.2. Atomicity](#)

[Section 2.3. Locking](#)

[Section 2.4. Guarding State with Locks](#)

[Section 2.5. Liveness and Performance](#)

[Chapter 3. Sharing Objects](#)

[Section 3.1. Visibility](#)

[Section 3.2. Publication and Escape](#)

[Section 3.3. Thread Confinement](#)

[Section 3.4. Immutability](#)

[Section 3.5. Safe Publication](#)

[Chapter 4. Composing Objects](#)

[Section 4.1. Designing a Thread-safe Class](#)

[Section 4.2. Instance Confinement](#)

[Section 4.3. Delegating Thread Safety](#)

[Section 4.4. Adding Functionality to Existing Thread-safe Classes](#)

[Section 4.5. Documenting Synchronization Policies](#)

[Chapter 5. Building Blocks](#)

[Section 5.1. Synchronized Collections](#)

[Section 5.2. Concurrent Collections](#)

[Section 5.3. Blocking Queues and the Producer-consumer Pattern](#)

[Section 5.4. Blocking and Interruptible Methods](#)
[Section 5.5. Synchronizers](#)
[Section 5.6. Building an Efficient, Scalable Result Cache](#)
[Summary of Part I](#)

[Part II: Structuring Concurrent Applications](#)

[Chapter 6. Task Execution](#)

[Section 6.1. Executing Tasks in Threads](#)
[Section 6.2. The Executor Framework](#)
[Section 6.3. Finding Exploitable Parallelism](#)
[Summary](#)

[Chapter 7. Cancellation and Shutdown](#)

[Section 7.1. Task Cancellation](#)
[Section 7.2. Stopping a Thread-based Service](#)
[Section 7.3. Handling Abnormal Thread Termination](#)
[Section 7.4. JVM Shutdown](#)
[Summary](#)

[Chapter 8. Applying Thread Pools](#)

[Section 8.1. Implicit Couplings Between Tasks and Execution Policies](#)
[Section 8.2. Sizing Thread Pools](#)
[Section 8.3. Configuring ThreadPoolExecutor](#)
[Section 8.4. Extending ThreadPoolExecutor](#)
[Section 8.5. Parallelizing Recursive Algorithms](#)
[Summary](#)

[Chapter 9. GUI Applications](#)

[Section 9.1. Why are GUIs Single-threaded?](#)
[Section 9.2. Short-running GUI Tasks](#)
[Section 9.3. Long-running GUI Tasks](#)
[Section 9.4. Shared Data Models](#)
[Section 9.5. Other Forms of Single-threaded Subsystems](#)
[Summary](#)

[Part III: Liveness, Performance, and Testing](#)

[Chapter 10. Avoiding Liveness Hazards](#)

[Section 10.1. Deadlock](#)
[Section 10.2. Avoiding and Diagnosing Deadlocks](#)
[Section 10.3. Other Liveness Hazards](#)
[Summary](#)

[Chapter 11. Performance and Scalability](#)

[Section 11.1. Thinking about Performance](#)
[Section 11.2. Amdahl's Law](#)
[Section 11.3. Costs Introduced by Threads](#)
[Section 11.4. Reducing Lock Contention](#)
[Section 11.5. Example: Comparing Map Performance](#)
[Section 11.6. Reducing Context Switch Overhead](#)
[Summary](#)

[Chapter 12. Testing Concurrent Programs](#)

[Section 12.1. Testing for Correctness](#)
[Section 12.2. Testing for Performance](#)

[Section 12.3. Avoiding Performance Testing Pitfalls](#)

[Section 12.4. Complementary Testing Approaches](#)

[Summary](#)

[Part IV: Advanced Topics](#)

[Chapter 13. Explicit Locks](#)

[Section 13.1. Lock and ReentrantLock](#)

[Section 13.2. Performance Considerations](#)

[Section 13.3. Fairness](#)

[Section 13.4. Choosing Between Synchronized and ReentrantLock](#)

[Section 13.5. Read-write Locks](#)

[Summary](#)

[Chapter 14. Building Custom Synchronizers](#)

[Section 14.1. Managing State Dependence](#)

[Section 14.2. Using Condition Queues](#)

[Section 14.3. Explicit Condition Objects](#)

[Section 14.4. Anatomy of a Synchronizer](#)

[Section 14.5. AbstractQueuedSynchronizer](#)

[Section 14.6. AQS in Java.util.concurrent Synchronizer Classes](#)

[Summary](#)

[Chapter 15. Atomic Variables and Nonblocking Synchronization](#)

[Section 15.1. Disadvantages of Locking](#)

[Section 15.2. Hardware Support for Concurrency](#)

[Section 15.3. Atomic Variable Classes](#)

[Section 15.4. Nonblocking Algorithms](#)

[Summary](#)

[Chapter 16. The Java Memory Model](#)

[Section 16.1. What is a Memory Model, and Why would I Want One?](#)

[Section 16.2. Publication](#)

[Section 16.3. Initialization Safety](#)

[Summary](#)

[Appendix A. Annotations for Concurrency](#)

[Section A.1. Class Annotations](#)

[Section A.2. Field and Method Annotations](#)

[Bibliography](#)

[Index](#)