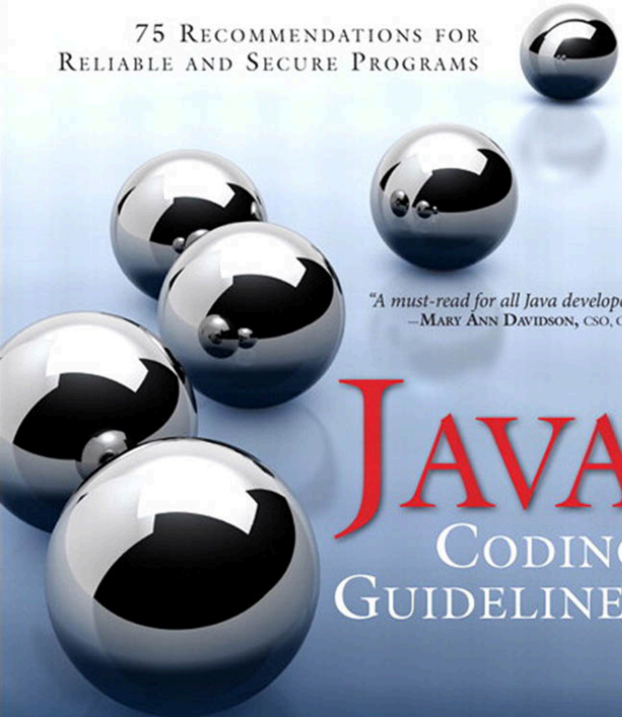
75 Recommendations for Reliable and Secure Programs



"A must-read for all Java developers."
—Mary Ann Davidson, CSO, Oracle

Java™ Coding Guidelines

Fred Long | Dhruv Mohindra | Robert C. Seacord
Dean F. Sutherland | David Svoboda

Foreword by James A. Gosling, Father of the Java Programming Language

Java™ Coding Guidelines

Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs

Table of Contents

Contents

Foreword

Preface

Acknowledgments

About the Authors

Chapter 1 Security

1. Limit the lifetime of sensitive data
2. Do not store unencrypted sensitive information on the client side
3. Provide sensitive mutable classes with unmodifiable wrappers
4. Ensure that security-sensitive methods are called with validated arguments
5. Prevent arbitrary file upload
6. Properly encode or escape output
7. Prevent code injection
8. Prevent XPath injection
9. Prevent LDAP injection
10. Do not use the clone() method to copy untrusted method parameters
11. Do not use Object.equals() to compare cryptographic keys
12. Do not use insecure or weak cryptographic algorithms

Pearson

Table of Contents

13. Store passwords using a hash function
14. Ensure that SecureRandom is properly seeded
15. Do not rely on methods that can be overridden by untrusted code
16. Avoid granting excess privileges
17. Minimize privileged code
18. Do not expose methods that use reduced-security checks to untrusted code
19. Define custom security permissions for fine-grained security
20. Create a secure sandbox using a security manager
21. Do not let untrusted code misuse privileges of callback methods

Chapter 2 Defensive Programming

22. Minimize the scope of variables
23. Minimize the scope of the @SuppressWarnings annotation
24. Minimize the accessibility of classes and their members
25. Document thread-safety and use annotations where applicable
26. Always provide feedback about the resulting value of a method
27. Identify files using multiple file attributes
28. Do not attach significance to the ordinal associated with an enum
29. Be aware of numeric promotion behavior
30. Enable compile-time type checking of variable arity parameter types
31. Do not apply public final to constants whose value might change in later releases
32. Avoid cyclic dependencies between packages
33. Prefer user-defined exceptions over more general exception types

Pearson

Table of Contents

34. Try to gracefully recover from system errors
35. Carefully design interfaces before releasing them
36. Write garbage collectionfriendly code

Chapter 3 Reliability

37. Do not shadow or obscure identifiers in subscopes
38. Do not declare more than one variable per declaration
39. Use meaningful symbolic constants to represent literal values in program logic
40. Properly encode relationships in constant definitions
41. Return an empty array or collection instead of a null value for methods that return an array or collection
42. Use exceptions only for exceptional conditions
43. Use a try-with-resources statement to safely handle closeable resources
44. Do not use assertions to verify the absence of runtime errors
45. Use the same type for the second and third operands in conditional expressions
46. Do not serialize direct handles to system resources
47. Prefer using iterators over enumerations
48. Do not use direct buffers for short-lived, infrequently used objects
49. Remove short-lived objects from long-lived container objects

Chapter 4 Program Understandability

50. Be careful using visually misleading identifiers and literals
51. Avoid ambiguous overloading of variable arity methods
52. Avoid in-band error indicators
53. Do not perform assignments in conditional expressions
54. Use braces for the body of an if, for, or while statement

Pearson

Table of Contents

55. Do not place a semicolon immediately following an if, for, or while condition
56. Finish every set of statements associated with a case label with a break statement
57. Avoid inadvertent wrapping of loop counters
58. Use parentheses for precedence of operation
59. Do not make assumptions about file creation
60. Convert integers to floating-point for floating-point operations
61. Ensure that the clone() method calls super.clone()
62. Use comments consistently and in a readable fashion
63. Detect and remove superfluous code and values
64. Strive for logical completeness
65. Avoid ambiguous or confusing uses of overloading

Chapter 5 Programmer Misconceptions

66. Do not assume that declaring a reference volatile guarantees safe publication of the members of the referenced object
67. Do not assume that the sleep(), yield(), or getState() methods provide synchronization semantics
68. Do not assume that the remainder operator always returns a nonnegative result for integral operands
69. Do not confuse abstract object equality with reference equality
70. Understand the differences between bitwise and logical operators
71. Understand how escape characters are interpreted when strings are loaded
72. Do not use overloaded methods to differentiate between runtime types

Table of Contents

73. Never confuse the immutability of a reference with that of the referenced object
74. Use the serialization methods writeUnshared() and readUnshared() with care
75. Do not attempt to help the garbage collector by setting local reference variables to null

Appendix A: Android

Glossary

References

Index

Pearson