ORACLE

The Java® Language Specification

Java SE 8 Edition

James Gosling

Bill Joy

Guy Steele

Gilad Bracha

Alex Buckley

2015-02-13

Legal Notice

Table of Contents

Preface to the Java SE 8 Edition
1. Introduction

- 1.1. Organization of the Specification
- 1.2. Example Programs
- 1.3. Notation
- 1.4. Relationship to Predefined Classes and Interfaces
- 1.5. Feedback
- 1.6. References

2. Grammars

- 2.1. Context-Free Grammars
- 2.2. The Lexical Grammar
- 2.3. The Syntactic Grammar
- 2.4. Grammar Notation

3. Lexical Structure

- 3.1. Unicode
- 3.2. Lexical Translations
- 3.3. Unicode Escapes
- 3.4. Line Terminators
- 3.5. Input Elements and Tokens
- 3.6. White Space
- 3.7. Comments

- 3.8. Identifiers
- 3.9. Keywords
- 3.10. Literals
    - 3.10.1. Integer Literals
    - 3.10.2. Floating-Point Literals
    - 3.10.3. Boolean Literals
    - 3.10.4. Character Literals
    - 3.10.5. String Literals
    - 3.10.6. Escape Sequences for Character and String Literals
    - 3.10.7. The Null Literal
- 3.11. Separators
- 3.12. Operators

4. Types, Values, and Variables

- 4.1. The Kinds of Types and Values
- 4.2. Primitive Types and Values
    - 4.2.1. Integral Types and Values
    - 4.2.2. Integer Operations
    - 4.2.3. Floating-Point Types, Formats, and Values
    - 4.2.4. Floating-Point Operations
    - 4.2.5. The `boolean` Type and boolean Values
- 4.3. Reference Types and Values
    - 4.3.1. Objects
    - 4.3.2. The Class `Object`
    - 4.3.3. The Class `String`
    - 4.3.4. When Reference Types Are the Same
- 4.4. Type Variables
- 4.5. Parameterized Types
    - 4.5.1. Type Arguments of Parameterized Types
    - 4.5.2. Members and Constructors of Parameterized Types
- 4.6. Type Erasure
- 4.7. Reifiable Types
- 4.8. Raw Types
- 4.9. Intersection Types
- 4.10. Subtyping
    - 4.10.1. Subtyping among Primitive Types
    - 4.10.2. Subtyping among Class and Interface Types
    - 4.10.3. Subtyping among Array Types
    - 4.10.4. Least Upper Bound
- 4.11. Where Types Are Used
- 4.12. Variables
    - 4.12.1. Variables of Primitive Type
    - 4.12.2. Variables of Reference Type
    - 4.12.3. Kinds of Variables
    - 4.12.4. `final` Variables
    - 4.12.5. Initial Values of Variables
    - 4.12.6. Types, Classes, and Interfaces

5. Conversions and Contexts

- 5.1. Kinds of Conversion
    - 5.1.1. Identity Conversion
    - 5.1.2. Widening Primitive Conversion
    - 5.1.3. Narrowing Primitive Conversion
    - 5.1.4. Widening and Narrowing Primitive Conversion
    - 5.1.5. Widening Reference Conversion

- 5.1.6. Narrowing Reference Conversion
- 5.1.7. Boxing Conversion
- 5.1.8. Unboxing Conversion
- 5.1.9. Unchecked Conversion
- 5.1.10. Capture Conversion
- 5.1.11. String Conversion
- 5.1.12. Forbidden Conversions
- 5.1.13. Value Set Conversion

5.2. Assignment Contexts
5.3. Invocation Contexts
5.4. String Contexts
5.5. Casting Contexts

- 5.5.1. Reference Type Casting
- 5.5.2. Checked Casts and Unchecked Casts
- 5.5.3. Checked Casts at Run Time

5.6. Numeric Contexts

- 5.6.1. Unary Numeric Promotion
- 6.6.2. Binary Numeric Promotion

6. Names

6.1. Declarations
6.2. Names and Identifiers
6.3. Scope of a Declaration
6.4. Shadowing and Obscuring

- 6.4.1. Shadowing
- 6.4.2. Obscuring

6.5. Determining the Meaning of a Name

- 6.5.1. Syntactic Classification of a Name According to Context
- 6.5.2. Reclassification of Contextually Ambiguous Names
- 6.5.3. Meaning of Package Names
  - 6.5.3.1. Simple Package Names
  - 6.5.3.2. Qualified Package Names
- 6.5.4. Meaning of *PackageOrTypeNames*
  - 6.5.4.1. Simple *PackageOrTypeNames*
  - 6.5.4.2. Qualified *PackageOrTypeNames*
- 6.5.5. Meaning of Type Names
  - 6.5.5.1. Simple Type Names
  - 6.5.5.2. Qualified Type Names
- 6.5.6. Meaning of Expression Names
  - 6.5.6.1. Simple Expression Names
  - 6.5.6.2. Qualified Expression Names
- 6.5.7. Meaning of Method Names
  - 6.5.7.1. Simple Method Names

6.6. Access Control

- 6.6.1. Determining Accessibility
- 6.6.2. Details on `protected` Access
  - 6.6.2.1. Access to a `protected` Member
  - 6.6.2.2. Qualified Access to a `protected` Constructor

6.7. Fully Qualified Names and Canonical Names

- 7. Packages
  - 7.1. Package Members
  - 7.2. Host Support for Packages
  - 7.3. Compilation Units
  - 7.4. Package Declarations
    - 7.4.1. Named Packages
    - 7.4.2. Unnamed Packages
    - 7.4.3. Observability of a Package
  - 7.5. Import Declarations
    - 7.5.1. Single-Type-Import Declarations
    - 7.5.2. Type-Import-on-Demand Declarations
    - 7.5.3. Single-Static-Import Declarations
    - 7.5.4. Static-Import-on-Demand Declarations
  - 7.6. Top Level Type Declarations
- 8. Classes
  - 8.1. Class Declarations
    - 8.1.1. Class Modifiers
      - 8.1.1.1. `abstract` Classes
      - 8.1.1.2. `final` Classes
      - 8.1.1.3. `strictfp` Classes
    - 8.1.2. Generic Classes and Type Parameters
    - 8.1.3. Inner Classes and Enclosing Instances
    - 8.1.4. Superclasses and Subclasses
    - 8.1.5. Superinterfaces
    - 8.1.6. Class Body and Member Declarations
  - 8.2. Class Members
  - 8.3. Field Declarations
    - 8.3.1. Field Modifiers
      - 8.3.1.1. `static` Fields
      - 8.3.1.2. `final` Fields
      - 8.3.1.3. `transient` Fields
      - 8.3.1.4. `volatile` Fields
    - 8.3.2. Field Initialization
    - 8.3.3. Forward References During Field Initialization
  - 8.4. Method Declarations
    - 8.4.1. Formal Parameters
    - 8.4.2. Method Signature
    - 8.4.3. Method Modifiers
      - 8.4.3.1. `abstract` Methods
      - 8.4.3.2. `static` Methods
      - 8.4.3.3. `final` Methods
      - 8.4.3.4. `native` Methods
      - 8.4.3.5. `strictfp` Methods
      - 8.4.3.6. `synchronized` Methods
    - 8.4.4. Generic Methods
    - 8.4.5. Method Result
    - 8.4.6. Method Throws
    - 8.4.7. Method Body
    - 8.4.8. Inheritance, Overriding, and Hiding
      - 8.4.8.1. Overriding (by Instance Methods)
      - 8.4.8.2. Hiding (by Class Methods)

- 8.4.8.3. Requirements in Overriding and Hiding
- 8.4.8.4. Inheriting Methods with Override-Equivalent Signatures
- 8.4.9. Overloading
- 8.5. Member Type Declarations
- 8.5.1. Static Member Type Declarations
- 8.6. Instance Initializers
- 8.7. Static Initializers
- 8.8. Constructor Declarations
- 8.8.1. Formal Parameters
- 8.8.2. Constructor Signature
- 8.8.3. Constructor Modifiers
- 8.8.4. Generic Constructors
- 8.8.5. Constructor Throws
- 8.8.6. The Type of a Constructor
- 8.8.7. Constructor Body
- 8.8.7.1. Explicit Constructor Invocations
- 8.8.8. Constructor Overloading
- 8.8.9. Default Constructor
- 8.8.10. Preventing Instantiation of a Class
- 8.9. Enum Types
- 8.9.1. Enum Constants
- 8.9.2. Enum Body Declarations
- 8.9.3. Enum Members
- 9. Interfaces
- 9.1. Interface Declarations
- 9.1.1. Interface Modifiers
- 9.1.1.1. `abstract` Interfaces
- 9.1.1.2. `strictfp` Interfaces
- 9.1.2. Generic Interfaces and Type Parameters
- 9.1.3. Superinterfaces and Subinterfaces
- 9.1.4. Interface Body and Member Declarations
- 9.2. Interface Members
- 9.3. Field (Constant) Declarations
- 9.3.1. Initialization of Fields in Interfaces
- 9.4. Method Declarations
- 9.4.1. Inheritance and Overriding
- 9.4.1.1. Overriding (by Instance Methods)
- 9.4.1.2. Requirements in Overriding
- 9.4.1.3. Inheriting Methods with Override-Equivalent Signatures
- 9.4.2. Overloading
- 9.4.3. Interface Method Body
- 9.5. Member Type Declarations
- 9.6. Annotation Types
- 9.6.1. Annotation Type Elements
- 9.6.2. Defaults for Annotation Type Elements
- 9.6.3. Repeatable Annotation Types
- 9.6.4. Predefined Annotation Types
- 9.6.4.1. `@Target`

- 9.6.4.2. @Retention
- 9.6.4.3. @Inherited
- 9.6.4.4. @Override
- 9.6.4.5. @SuppressWarnings
- 9.6.4.6. @Deprecated
- 9.6.4.7. @SafeVarargs
- 9.6.4.8. @Repeatable
- 9.6.4.9. @FunctionalInterface

9.7. Annotations

- 9.7.1. Normal Annotations
- 9.7.2. Marker Annotations
- 9.7.3. Single-Element Annotations
- 9.7.4. Where Annotations May Appear
- 9.7.5. Multiple Annotations of the Same Type

9.8. Functional Interfaces
9.9. Function Types

10. Arrays

- 10.1. Array Types
- 10.2. Array Variables
- 10.3. Array Creation
- 10.4. Array Access
- 10.5. Array Store Exception
- 10.6. Array Initializers
- 10.7. Array Members
- 10.8. Class Objects for Arrays
- 10.9. An Array of Characters Is Not a String

11. Exceptions

11.1. The Kinds and Causes of Exceptions

- 11.1.1. The Kinds of Exceptions
- 11.1.2. The Causes of Exceptions
- 11.1.3. Asynchronous Exceptions

11.2. Compile-Time Checking of Exceptions

- 11.2.1. Exception Analysis of Expressions
- 11.2.2. Exception Analysis of Statements
- 11.2.3. Exception Checking

11.3. Run-Time Handling of an Exception

12. Execution

12.1. Java Virtual Machine Startup

- 12.1.1. Load the Class Test
- 12.1.2. Link Test: Verify, Prepare, (Optionally) Resolve
- 12.1.3. Initialize Test: Execute Initializers
- 12.1.4. Invoke Test.main

12.2. Loading of Classes and Interfaces

- 12.2.1. The Loading Process

12.3. Linking of Classes and Interfaces

- 12.3.1. Verification of the Binary Representation
- 12.3.2. Preparation of a Class or Interface Type
- 12.3.3. Resolution of Symbolic References

12.4. Initialization of Classes and Interfaces

- 12.4.1. When Initialization Occurs

- 12.4.2. Detailed Initialization Procedure
- 12.5. Creation of New Class Instances
- 12.6. Finalization of Class Instances
  - 12.6.1. Implementing Finalization
  - 12.6.2. Interaction with the Memory Model
- 12.7. Unloading of Classes and Interfaces
- 12.8. Program Exit

13. Binary Compatibility
- 13.1. The Form of a Binary
- 13.2. What Binary Compatibility Is and Is Not
- 13.3. Evolution of Packages
- 13.4. Evolution of Classes
  - 13.4.1. `abstract` Classes
  - 13.4.2. `final` Classes
  - 13.4.3. `public` Classes
  - 13.4.4. Superclasses and Superinterfaces
  - 13.4.5. Class Type Parameters
  - 13.4.6. Class Body and Member Declarations
  - 13.4.7. Access to Members and Constructors
  - 13.4.8. Field Declarations
  - 13.4.9. `final` Fields and `static` Constant Variables
  - 13.4.10. `static` Fields
  - 13.4.11. `transient` Fields
  - 13.4.12. Method and Constructor Declarations
  - 13.4.13. Method and Constructor Type Parameters
  - 13.4.14. Method and Constructor Formal Parameters
  - 13.4.15. Method Result Type
  - 13.4.16. `abstract` Methods
  - 13.4.17. `final` Methods
  - 13.4.18. `native` Methods
  - 13.4.19. `static` Methods
  - 13.4.20. `synchronized` Methods
  - 13.4.21. Method and Constructor Throws
  - 13.4.22. Method and Constructor Body
  - 13.4.23. Method and Constructor Overloading
  - 13.4.24. Method Overriding
  - 13.4.25. Static Initializers
  - 13.4.26. Evolution of Enums
- 13.5. Evolution of Interfaces
  - 13.5.1. `public` Interfaces
  - 13.5.2. Superinterfaces
  - 13.5.3. Interface Members
  - 13.5.4. Interface Type Parameters
  - 13.5.5. Field Declarations
  - 13.5.6. Interface Method Declarations
  - 13.5.7. Evolution of Annotation Types

14. Blocks and Statements
- 14.1. Normal and Abrupt Completion of Statements
- 14.2. Blocks
- 14.3. Local Class Declarations
- 14.4. Local Variable Declaration Statements
  - 14.4.1. Local Variable Declarators and Types
  - 14.4.2. Execution of Local Variable Declarations
- 14.5. Statements
- 14.6. The Empty Statement
- 14.7. Labeled Statements

- 14.8. Expression Statements
- 14.9. The `if` Statement
  - 14.9.1. The `if-then` Statement
  - 14.9.2. The `if-then-else` Statement
- 14.10. The `assert` Statement
- 14.11. The `switch` Statement
- 14.12. The `while` Statement
  - 14.12.1. Abrupt Completion of `while` Statement
- 14.13. The `do` Statement
  - 14.13.1. Abrupt Completion of `do` Statement
- 14.14. The `for` Statement
  - 14.14.1. The basic `for` Statement
    - 14.14.1.1. Initialization of `for` Statement
    - 14.14.1.2. Iteration of `for` Statement
    - 14.14.1.3. Abrupt Completion of `for` Statement
  - 14.14.2. The enhanced `for` statement
- 14.15. The `break` Statement
- 14.16. The `continue` Statement
- 14.17. The `return` Statement
- 14.18. The `throw` Statement
- 14.19. The `synchronized` Statement
- 14.20. The `try` statement
  - 14.20.1. Execution of `try-catch`
  - 14.20.2. Execution of `try-finally` and `try-catch-finally`
  - 14.20.3. `try`-with-resources
    - 14.20.3.1. Basic `try`-with-resources
    - 14.20.3.2. Extended `try`-with-resources
- 14.21. Unreachable Statements

15. Expressions
- 15.1. Evaluation, Denotation, and Result
- 15.2. Forms of Expressions
- 15.3. Type of an Expression
- 15.4. FP-strict Expressions
- 15.5. Expressions and Run-Time Checks
- 15.6. Normal and Abrupt Completion of Evaluation
- 15.7. Evaluation Order
  - 15.7.1. Evaluate Left-Hand Operand First
  - 15.7.2. Evaluate Operands before Operation
  - 15.7.3. Evaluation Respects Parentheses and Precedence
  - 15.7.4. Argument Lists are Evaluated Left-to-Right
  - 15.7.5. Evaluation Order for Other Expressions
- 15.8. Primary Expressions
  - 15.8.1. Lexical Literals
  - 15.8.2. Class Literals
  - 15.8.3. `this`
  - 15.8.4. Qualified `this`
  - 15.8.5. Parenthesized Expressions
- 15.9. Class Instance Creation Expressions
  - 15.9.1. Determining the Class being Instantiated
  - 15.9.2. Determining Enclosing Instances

15.9.3. Choosing the Constructor and its Arguments
15.9.4. Run-Time Evaluation of Class Instance Creation Expressions
15.9.5. Anonymous Class Declarations

15.9.5.1. Anonymous Constructors

15.10. Array Creation and Access Expressions

15.10.1. Array Creation Expressions
15.10.2. Run-Time Evaluation of Array Creation Expressions
15.10.3. Array Access Expressions
15.10.4. Run-Time Evaluation of Array Access Expressions

15.11. Field Access Expressions

15.11.1. Field Access Using a Primary
15.11.2. Accessing Superclass Members using `super`

15.12. Method Invocation Expressions

15.12.1. Compile-Time Step 1: Determine Class or Interface to Search
15.12.2. Compile-Time Step 2: Determine Method Signature

15.12.2.1. Identify Potentially Applicable Methods
15.12.2.2. Phase 1: Identify Matching Arity Methods Applicable by Strict Invocation
15.12.2.3. Phase 2: Identify Matching Arity Methods Applicable by Loose Invocation
15.12.2.4. Phase 3: Identify Methods Applicable by Variable Arity Invocation
15.12.2.5. Choosing the Most Specific Method
15.12.2.6. Method Invocation Type

15.12.3. Compile-Time Step 3: Is the Chosen Method Appropriate?
15.12.4. Run-Time Evaluation of Method Invocation

15.12.4.1. Compute Target Reference (If Necessary)
15.12.4.2. Evaluate Arguments
15.12.4.3. Check Accessibility of Type and Method
15.12.4.4. Locate Method to Invoke
15.12.4.5. Create Frame, Synchronize, Transfer Control

15.13. Method Reference Expressions

15.13.1. Compile-Time Declaration of a Method Reference
15.13.2. Type of a Method Reference
15.13.3. Run-Time Evaluation of Method References

15.14. Postfix Expressions

15.14.1. Expression Names
15.14.2. Postfix Increment Operator `++`
15.14.3. Postfix Decrement Operator `--`

15.15. Unary Operators

15.15.1. Prefix Increment Operator `++`
15.15.2. Prefix Decrement Operator `--`
15.15.3. Unary Plus Operator `+`
15.15.4. Unary Minus Operator `-`
15.15.5. Bitwise Complement Operator `~`
15.15.6. Logical Complement Operator `!`

15.16. Cast Expressions
15.17. Multiplicative Operators

15.17.1. Multiplication Operator `*`
15.17.2. Division Operator `/`
15.17.3. Remainder Operator `%`

15.18. Additive Operators

15.18.1. String Concatenation Operator `+`

15.18.2. Additive Operators (+ and -) for Numeric Types

15.19. Shift Operators
15.20. Relational Operators

15.20.1. Numerical Comparison Operators <, <=, >, and >=
15.20.2. Type Comparison Operator instanceof

15.21. Equality Operators

15.21.1. Numerical Equality Operators == and !=
15.21.2. Boolean Equality Operators == and !=
15.21.3. Reference Equality Operators == and !=

15.22. Bitwise and Logical Operators

15.22.1. Integer Bitwise Operators &, ^, and |
15.22.2. Boolean Logical Operators &, ^, and |

15.23. Conditional-And Operator &&
15.24. Conditional-Or Operator ||
15.25. Conditional Operator ? :

15.25.1. Boolean Conditional Expressions
15.25.2. Numeric Conditional Expressions
15.25.3. Reference Conditional Expressions

15.26. Assignment Operators

15.26.1. Simple Assignment Operator =
15.26.2. Compound Assignment Operators

15.27. Lambda Expressions

15.27.1. Lambda Parameters
15.27.2. Lambda Body
15.27.3. Type of a Lambda Expression
15.27.4. Run-Time Evaluation of Lambda Expressions

15.28. Constant Expressions

16. Definite Assignment

16.1. Definite Assignment and Expressions

16.1.1. Boolean Constant Expressions
16.1.2. Conditional-And Operator &&
16.1.3. Conditional-Or Operator ||
16.1.4. Logical Complement Operator !
16.1.5. Conditional Operator ? :
16.1.6. Conditional Operator ? :
16.1.7. Other Expressions of Type boolean
16.1.8. Assignment Expressions
16.1.9. Operators ++ and --
16.1.10. Other Expressions

16.2. Definite Assignment and Statements

16.2.1. Empty Statements
16.2.2. Blocks
16.2.3. Local Class Declaration Statements
16.2.4. Local Variable Declaration Statements
16.2.5. Labeled Statements
16.2.6. Expression Statements
16.2.7. if Statements
16.2.8. assert Statements
16.2.9. switch Statements
16.2.10. while Statements
16.2.11. do Statements

- 16.2.12. `for` Statements
  - 16.2.12.1. Initialization Part of `for` Statement
  - 16.2.12.2. Incrementation Part of `for` Statement
- 16.2.13. `break`, `continue`, `return`, and `throw` Statements
- 16.2.14. `synchronized` Statements
- 16.2.15. `try` Statements

16.3. Definite Assignment and Parameters
16.4. Definite Assignment and Array Initializers
16.5. Definite Assignment and Enum Constants
16.6. Definite Assignment and Anonymous Classes
16.7. Definite Assignment and Member Types
16.8. Definite Assignment and Static Initializers
16.9. Definite Assignment, Constructors, and Instance Initializers

17. Threads and Locks

17.1. Synchronization
17.2. Wait Sets and Notification

- 17.2.1. Wait
- 17.2.2. Notification
- 17.2.3. Interruptions
- 17.2.4. Interactions of Waits, Notification, and Interruption

17.3. Sleep and Yield
17.4. Memory Model

- 17.4.1. Shared Variables
- 17.4.2. Actions
- 17.4.3. Programs and Program Order
- 17.4.4. Synchronization Order
- 17.4.5. Happens-before Order
- 17.4.6. Executions
- 17.4.7. Well-Formed Executions
- 17.4.8. Executions and Causality Requirements
- 17.4.9. Observable Behavior and Nonterminating Executions

17.5. `final` Field Semantics

- 17.5.1. Semantics of `final` Fields
- 17.5.2. Reading `final` Fields During Construction
- 17.5.3. Subsequent Modification of `final` Fields
- 17.5.4. Write-Protected Fields

17.6. Word Tearing
17.7. Non-Atomic Treatment of `double` and `long`

18. Type Inference

18.1. Concepts and Notation

- 18.1.1. Inference Variables
- 18.1.2. Constraint Formulas
- 18.1.3. Bounds

18.2. Reduction

- 18.2.1. Expression Compatibility Constraints
- 18.2.2. Type Compatibility Constraints
- 18.2.3. Subtyping Constraints
- 18.2.4. Type Equality Constraints
- 18.2.5. Checked Exception Constraints

18.3. Incorporation

- 18.3.1. Complementary Pairs of Bounds

18.3.2. Bounds Involving Capture Conversion

18.4. Resolution
18.5. Uses of Inference

18.5.1. Invocation Applicability Inference
18.5.2. Invocation Type Inference
18.5.3. Functional Interface Parameterization Inference
18.5.4. More Specific Method Inference

19. Syntax
Index
A. Limited License Grant

List of Examples

3.10.5-1. String Literals
4.2.2-1. Integer Operations
4.2.4-1. Floating-point Operations
4.3.1-1. Object Creation
4.3.1-2. Primitive and Reference Identity
4.4-1. Members of a Type Variable
4.5.1-1. Unbounded Wildcards
4.5.1-2. Bounded Wildcards
4.8-1. Raw Types
4.8-2. Raw Types and Inheritance
4.11-1. Usage of a Type
4.12.3-1. Different Kinds of Variables
4.12.4-1. Final Variables
4.12.5-1. Initial Values of Variables
4.12.6-1. Type of a Variable versus Class of an Object
5.0-1. Conversions at Compile Time and Run Time
5.0-2. Conversions In Various Contexts
5.1.2-1. Widening Primitive Conversion
5.1.3-1. Narrowing Primitive Conversion
5.1.3-2. Narrowing Primitive Conversions that lose information
5.2-1. Assignment Conversion for Primitive Types
5.2-2. Assignment Conversion for Reference Types
5.2-3. Assignment Conversion for Array Types
5.5.1-1. Casting Conversion for Reference Types
5.5.1-2. Casting Conversion for Array Types
5.5.3-1. Incompatible Types at Run Time
5.6.1-1. Unary Numeric Promotion
5.6.2-1. Binary Numeric Promotion
6.1-1. Unique Package Names
6.1-2. Descriptive Class Names
6.1-3. Conventional Type Variable Names
6.3-1. Scope of Type Declarations
6.3-2. Scope of Local Variable Declarations
6.4-1. Attempted Shadowing Of A Local Variable
6.4.1-1. Shadowing of a Field Declaration by a Local Variable Declaration
6.4.1-2. Shadowing of a Type Declaration by Another Type Declaration
6.5.2-1. Reclassification of Contextually Ambiguous Names
6.5.5.2-1. Qualified Type Names
6.5.6.1-1. Simple Expression Names
6.5.6.2-1. Qualified Expression Names
6.5.6.2-2. Qualifying an Expression with a Type Name
6.5.7.1-1. Simple Method Names and Visibility
6.6-1. Access Control
6.6-2. Access to `public` Fields, Methods, and Constructors
6.6-3. Access to `public` and Non-`public` Classes
6.6-4. Access to Package-Access Fields, Methods, and Constructors
6.6-5. Access to `private` Fields, Methods, and Constructors
6.6.2-1. Access to `protected` Fields, Methods, and Constructors
6.7-1. Fully Qualified Names

6.7-2. Fully Qualified Names v. Canonical Name
7.4.2-1.
7.5.1-1. Single-Type-Import
7.5.1-2. Duplicate Type Declarations
7.5.1-3. No Import of a Subpackage
7.5.1-4. Importing a Type Name that is also a Package Name
7.5.2-1. Type-Import-on-Demand
7.6-1. Conflicting Top Level Type Declarations
7.6-2. Scope of Top Level Types
7.6-3. Fully Qualified Names
8.1.1.1-1. Abstract Class Declaration
8.1.1.1-2. Abstract Class Declaration that Prohibits Subclasses
8.1.2-1. Mutually Recursive Type Variable Bounds
8.1.2-2. Nested Generic Classes
8.1.3-1. Inner Class Declarations and Static Members
8.1.3-2. Inner Class Declarations
8.1.4-1. Direct Superclasses and Subclasses
8.1.4-2. Superclasses and Subclasses
8.1.4-3. Class Depends on Itself
8.1.5-1. Illegal Superinterfaces
8.1.5-2. Superinterfaces
8.1.5-3. Illegal Multiple Inheritance of an Interface
8.1.5-3. Implementing Methods of a Superinterface
8.2-1. Use of Class Members
8.2-2. Inheritance of Class Members with Package Access
8.2-3. Inheritance of `public` and `protected` Class Members
8.2-4. Inheritance of `private` Class Members
8.2-5. Accessing Members of Inaccessible Classes
8.3-1. Multiply Inherited Fields
8.3-2. Re-inheritance of Fields
8.3.1.1-1. `static` Fields
8.3.1.1-2. Hiding of Class Variables
8.3.1.1-3. Hiding of Instance Variables
8.3.1.3-1. Persistence of `transient` Fields
8.3.1.4-1. `volatile` Fields
8.3.2-1. Field Initialization
8.3.2-2. Forward Reference to a Class Variable
8.3.3-1. Restrictions on Field Initialization
8.4.2-1. Override-Equivalent Signatures
8.4.3.1-1. Abstract/Abstract Method Overriding
8.4.3.1-2. Abstract/Non-Abstract Overriding
8.4.3.6-1. `synchronized` Monitors
8.4.3.6-2. `synchronized` Methods
8.4.6-1. Type Variables as Thrown Exception Types
8.4.8.1-1. Overriding
8.4.8.1-2. Overriding
8.4.8.2-1. Invocation of Hidden Class Methods
8.4.8.3-1. Covariant Return Types
8.4.8.3-2. Unchecked Warning from Return Type
8.4.8.3-3. Incorrect Overriding because of `throws`
8.4.8.3-4. Erasure Affects Overriding
8.4.9-1. Overloading
8.4.9-2. Overloading, Overriding, and Hiding
8.8-1. Constructor Declarations
8.8.7-1. Constructor Bodies
8.8.7.1-1. Restrictions on Explicit Constructor Invocation Statements
8.8.7.1-2. Qualified Superclass Constructor Invocation
8.8.9-1. Default Constructors
8.8.9-2. Accessibility of Constructors v. Classes
8.8.10-1. Preventing Instantiation via Constructor Accessibility
8.9.2-1. Enum Body Declarations
8.9.2-2. Restriction On Enum Constant Self-Reference
8.9.3-1. Iterating Over Enum Constants With An Enhanced `for` Loop
8.9.3-2. Switching Over Enum Constants
8.9.3-3. Enum Constants with Class Bodies

8.9.3-4. Multiple Enum Types
9.3-1. Ambiguous Inherited Fields
9.3-2. Multiply Inherited Fields
9.3.1-1. Forward Reference to a Field
9.4.2-1. Overloading an `abstract` Method Declaration
9.6.1-1. Annotation Type Declaration
9.6.1-2. Marker Annotation Type Declaration
9.6.1-3. Single-Element Annotation Type Declarations
9.6.2-1. Annotation Type Declaration With Default Values
9.6.3-1. Ill-formed Containing Annotation Type
9.6.3-2. Restricting Where Annotations May Repeat
9.6.3-3. A Repeatable Containing Annotation Type
9.7.1-1. Normal Annotations
9.7.2-1. Marker Annotations
9.7.3-1. Single-Element Annotations
9.8-1. Functional Interfaces
9.8-2. Functional Interfaces and Erasure
9.8-3. Generic Functional Interfaces
9.9-1. Function Types
9.9-2. Generic Function Types
10.2-1. Declarations of Array Variables
10.2-2. Array Variables and Array Types
10.4-1. Array Access
10.5-1. `ArrayStoreException`
10.6-1. Array Initializers
10.7-1. Arrays Are Cloneable
10.7-2. Shared Subarrays After A Clone
10.8-1. `Class` Object Of Array
10.8-2. Array `Class` Objects Are Shared
11.2.3-1. Catching Checked Exceptions
11.3-1. Throwing and Catching Exceptions
12.4.1-1. Superclasses Are Initialized Before Subclasses
12.4.1-2. Only The Class That Declares `static` Field Is Initialized
12.4.1-3. Interface Initialization Does Not Initialize Superinterfaces
12.5-1. Evaluation of Instance Creation
12.5-2. Dynamic Dispatch During Instance Creation
13.4.4-1. Changing A Superclass
13.4.6-1. Changing A Class Body
13.4.6-2. Changing A Superclass
13.4.7-1. Changing Accessibility
13.4.8-1. Adding A Field Declaration
13.4.9-1. Changing A Variable To Be `final`
13.4.9-2. Conditional Compilation
13.4.16-1. Changing A Method To Be `abstract`
13.4.17-1. Changing A Method To Be `final`
13.4.23-1. Adding An Overloaded Method
13.5.3-1. Deleting An Interface Member
13.5.6-1. Adding A Default Method
14.3-1. Local Class Declarations
14.7-1. Labels and Identifiers
14.11-1. Fall-Through in the `switch` Statement
14.13-1. The `do` Statement
14.14-1. Enhanced `for` And Arrays
14.14-2. Enhanced `for` And Unboxing Conversion
14.15-1. The `break` Statement
14.16-1. The `continue` Statement
14.19-1. The `synchronized` Statement
14.20.1-1. Catching An Exception
14.20.2-1. Handling An Uncaught Exception With `finally`
15.7.1-1. Left-Hand Operand Is Evaluated First
15.7.1-2. Implicit Left-Hand Operand In Operator Of Compound Assigment
15.7.1-3. Abrupt Completion of Evaluation of the Left-Hand Operand
15.7.2-1. Evaluation of Operands Before Operation
15.7.4-1. Evaluation Order At Method Invocation
15.7.4-2. Abrupt Completion of Argument Expression

15.8.3-1. The this Expression
15.9.4-1. Evaluation Order and Out-Of-Memory Detection
15.10.2-1. Array Creation Evaluation
15.10.2-2. Multi-Dimensional Array Creation
15.10.2-3. OutOfMemoryError and Dimension Expression Evaluation
15.10.4-1. Array Reference Is Evaluated First
15.10.4-2. Abrupt Completion of Array Reference Evaluation
15.10.4-3. null Array Reference
15.11.1-1. Static Binding for Field Access
15.11.1-2. Receiver Variable Is Irrelevant For static Field Access
15.11.2-1. The super Expression
15.12.2-1. Method Applicability
15.12.2-2. Return Type Not Considered During Method Selection
15.12.2-3. Choosing The Most Specific Method
15.12.4.1-1. Target References and static Methods
15.12.4.1-2. Evaluation Order During Method Invocation
15.12.4.4-1. Overriding and Method Invocation
15.12.4.4-2. Method Invocation Using super
15.12.4.5-1. Invoked Method Signature Has Different Erasure Than Compile-Time Method Signature
15.17.3-1. Integer Remainder Operator
15.17.3-2. Floating-Point Remainder Operator
15.18.1-1. String Concatenation
15.18.1-2. String Concatenation and Conditionals
15.20.2-1. The instanceof Operator
15.26.1-1. Simple Assignment To An Array Component
15.26.2-1. Compound Assignment To An Array Component
15.26.2-2. Value Of Left-Hand Side Of Compound Assignment Is Saved Before Evaluation Of Right-Hand Side
15.28-1. Constant Expressions
16-1. Definite Assignment Considers Structure of Statements and Expressions
16-2. Definite Assignment Does Not Consider Values of Expressions
16-3. Definite Unassignment
17.4-1. Incorrectly Synchronized Programs May Exhibit Surprising Behavior
17.4.5-1. Happens-before Consistency
17.4.8-1. Happens-before Consistency Is Not Sufficient
17.5-1. final Fields In The Java Memory Model
17.5-2. final Fields For Security
17.5.3-1. Aggressive Optimization of final Fields
17.6-1. Detection of Word Tearing

Next
Preface to the Java SE 8 Edition

Legal Notice