

Programming in Java™

© 2001, 2005 Rex Jaeschke. All rights reserved.

Once the reader has read the information provided in each chapter, and solved the related programming exercises, he or she should be able to do the following:

1. The Basics

- Understand the use of white space, comments, and blocks, and to identify individual source tokens.
- Understand Java's compilation model.
- For user-defined names, know the rules regarding their spelling, case sensitivity, and length of significance.
- Know how to use the basic output facilities to write to the screen.
- Understand the size and value range of the arithmetic data types.
- Use the boolean and enum data types.
- Know how to write simple expressions and statements.
- Know how to write literals and be able to tell their type.
- Know how to define and initialize local variables.
- Understand and apply operator precedence and associativity.
- Know when and how to use implicit and explicit type conversion.
- Have a basic understanding on how to break a program into multiple classes, and how to communicate between them.
- Know how and when to use the public and private modifiers.
- Know how to define final variables.
- Know how to use methods in the standard library classes, such as Character, Integer, Double, and Math.

2. Looping and Branching

- Use the while, for, for each, and do-while looping constructs.
- Make decisions using if-else and switch.
- Understand when and how to use the branching statements break and continue.
- Understand the purpose of, and problems associated with, the null statement.
- Start to appreciate certain aspects of programming style with respect to white space usage and indenting.
- Understand the prefix and postfix increment and decrement operators.
- Know the family of relational and equality operators.

- Understand that assignment is an operator, and the implications thereof, especially with regard to embedded assignment and the potential for confusion with equality.
- Know about the basic compound-assignment operators.

3. Methods

- Know how to pass arguments to methods by value.
- Know how to return a value from a method.
- Understand method definitions and their role in allowing the compiler to check method calls for correctness and to cause implicit conversion.
- Have a basic understanding of recursion.
- Be able to overload methods.

4. References, Strings, and Arrays

- Understand that a reference variable points to an object rather than containing that object's value directly, and that multiple reference variables can point to the same object.
- Understand the purpose of a null reference.
- Know how to pass reference arguments to and return them from a method.
- Know how to define and perform basic operations using the type String.
- Know how to allocate memory dynamically using new.
- Have a basic understanding of a reference count and that garbage collection is automatic.
- Define, initialize, and use single dimensional arrays and arrays of arrays.
- Understand that each dimension's elements start at index 0 and that each dimension has a read-only length field.
- Know how arrays are stored in memory.
- Know how to deal with command-line arguments passed to main.
- Understand why System.arraycopy is needed and how to use it.
- Know how to define and use variable-length argument lists.
- Know why the primitive wrapper classes exist and how to use them.
- Have a basic understanding of the StringBuffer class.
- Understand and be able to use static initialization blocks.

5. Classes

- Understand the purpose of data hiding and encapsulation.
- Be able to use effectively the modifiers public and private.
- Be able to implement a simple version of equals and toString for a class.
- Know how to have one constructor call another for the same class.
- Know that the way in which an object is represented internally is not necessarily related to the way in which programmers see it externally.
- Know the type of this, and what it refers to.
- Understand the difference between instance and class data.
- Understand and be able to use instance initialization blocks.
- Be able to define simple user-defined types using object-oriented constructs.
- Know that class definitions can nest.

6. Inheritance

- Know how and why one would want to use inheritance.
- Understand the difference between containment and inheritance.
- At the design stage, be able to study a set of class descriptions and determine if those classes are unrelated, related by inheritance, or related by containment.
- Know how and when to use the keyword super.
- Know how and why a class should be made abstract.
- Know how and why a method should be made abstract.
- Know when and how to use the protected access specifier.
- Understand the advantage of having Object as the ultimate superclass.
- Know when and how to use the instanceof operator.
- Understanding boxing and unboxing.
- Have a basic understanding of interfaces.
- Be able to use generic methods and types.

7. Exception Handling

- Understand how the traditional approaches to dealing with extraordinary errors are limited and messy in general and even more so in an object-oriented environment.
- Know how to catch system exceptions using try and catch blocks, and how to recover from such exceptions where possible.
- Be able to throw an exception of a given type.
- Understand that an exception type could be a full-blown class using all the facilities we've learned so far.
- Understand the throws clause.
- Understand the utility of, and issues relating to, a family of derived exception types.

8. Packages

- Understand how namespace pollution is a problem in projects involving multiple subsystems and development groups.
- Know how to import names from packages, and how to use names within those packages without importing them.
- Be able to define packages.

9. Input and Output

- Understand the concept of streams.
- Know how to detect exceptions that occur during I/O.
- Be able to perform basic I/O operations to/from the screen, a file, an array, and a string.
- Understand the difference between formatted and unformatted I/O.
- Know that one can random move about a file, saving and restoring file positions at will.
- Know that basic operations can be performed on files and directories.