**3**

**Programming with Java Operators and Strings**

**Understand Fundamental Operators**

Java operators are used to return a result from an expression using one, two, or three operands. Operands are the values placed to the right or left side of the operators. Prefix- and postfix-increment and prefix- and postfix-decrement operators use one operand. The conditional ternary operator (?:) uses three operands. All other operators use two operands.

The following topics will be covered in these pages:

* Assignment operators
* Arithmetic operators
* Relational operators
* Logical operators
* Operator precedence

**Assignment Operators**

Assignment operators are used to assign values to variables.

|  |  |
| --- | --- |
| = | Assignment operator |

**Compound Assignment Operators**

|  |  |
| --- | --- |
| + = | Assignment by addition operator |
| * = | Assignment by subtraction operator |

Although the use of compound assignment operators cuts down on keystrokes, it is generally good practice to use the “longhand” approach since the code is clearly more readable.

*It is common to represent assignments in pseudo-code with the colon and equal sign characters (for example, A := 20). Notice that := looks similar to +=, -=, and other Java assignment operators such as \*=, /=, and %=. Be aware, however, that the pseudo-code assignment representation (:=) is not a Java assignment operator, and if you see it in any Java code, it will not compile.*

**Arithmetic Operators**

**Basic Arithmetic Operators**

|  |  |
| --- | --- |
| + | Addition (sum) operator |
| - | Subtraction (difference) operator |
| \* | Multiplication (product) operator |
| / | Division (quotient) operator |
| % | Modulus (remainder) operator |

**Prefix-Increment, Postfix-Increment, Prefix-Decrement, and Postfix-Decrement Operators**

|  |  |
| --- | --- |
| ++x | Prefix-increment operator |
| --x | Prefix-decrement operator |
| x++ | Postfix-increment operator |
| x-- | Postfix-decrement operator |

Prefix-increment and prefix-decrement operators provide a shorthand way of incrementing and decrementing the variable by 1. Rather than creating an expression as y=x+1, you could write y=++x. Similarly, you could replace the expression y=x-1 with y=--x. This works because the execution of the prefix operators occurs on the operand prior to the evaluation of the whole expression. Postfix-increment and postfix-decrement characters execute the postfix operators after the expression has been evaluated. Therefore, y = x++ would equate to y=x followed by x=x+1. And y = x-- would equate to y=x followed by x=x-1.

Note that y=++x is not exactly equivalent to y=x+1, because the value of x changes in the former but not in the latter. This is the same for y=--x and y=x-1.

The prefix-increment operator increments a value by 1 before an expression has been evaluated.

|  |
| --- |
| int x = 10;  int y = ++x ;  System.out.println("x=" + x + ", y=" + y); // x= 11, y= 11 |

The postfix-increment operator increments a value by 1 after an expression has been evaluated.

|  |
| --- |
| int x = 10;  int y = x++ ;  System.out.println("x=" + x + ", y=" + y); // x= 11, y= 10 |

The prefix-decrement operator decrements a value by 1 before an expression has been evaluated.

|  |
| --- |
| int x = 10;  int y = --x ;  System.out.println("x=" + x + ", y=" + y); // x= 9, y= 9 |

The postfix-decrement operator decrements a value by 1 after an expression has been evaluated.

|  |
| --- |
| int x = 10;  int y = x-- ;  System.out.println("x=" + x + ", y=" + y); // x= 9, y= 10 |

**Relational Operators**

**Basic Relational Operators**

|  |  |
| --- | --- |
| < | Less than operator |
| <= | Less than or equal to operator |
| > | Greater than operator |
| >= | Greater than or equal to operator |

These operators are used to compare integers, floating points, and characters.

Remember that characters (that is, char primitives) accept integers (within the valid 16-bit unsigned range), hexadecimal, octal, and character literals.

|  |
| --- |
| boolean b9 = 'A' < 'B'; //Character literals  boolean b10 = '\u0041' < '\u0042'; //Unicode literals  boolean b11 = 65 < 66; //Integer literals that fit in a char  boolean b12 = '\101' < '\102'; //Octal literals |

The relationship between floating points can also be tested:

|  |
| --- |
| boolean b14 = 9.00D < 9.50D; // Floating points with D postfixes:  boolean b15 = 9.00d < 9.50d; // Floating points with d postfixes  boolean b16 = 9.00F < 9.50F; // Floating points with d postfixes  boolean b17 = 9.00f < 9.50f; // Floating points with f postfixes  boolean b18 = (double) 9 < (double) 10; // Integers with specific casts  boolean b19 = (float) 9 < (float) 10; // Integers with specific casts  boolean b20 = 9 < 10; // Integers that fit into floating points  boolean b21 = (9 < 10f);  boolean b22 = (float) 11 < 12; |