Appendix B Precedence Rules

Precedence rules establish the order of operations when an expression is evaluated. For example, in 3 + 4 * 4, is the answer 19 or 28? It depends on whether you multiply or add first. Normal precedence rules dictate that multiplication is done before addition. Precedence can be overridden using parentheses. For example, (3 + 4) * 4 means that the addition should be performed before the multiplication, yielding 28.

Precedence of Java Operators

Table B-1 lists all of the Java operators in order from the highest precedence (operators that are used first) to the lowest (operators that are used last). Sometimes different operators have the same precedence. In this case, they are listed in indented groups. For example, all of the multiplicative operators (`*`, `/`, `%`) have the same precedence.

When two operators with the same precedence appear together, the operators are performed left to right. That is, 3 * 4 / 6 is the same as (3 * 4) / 6. The only exception is assignment. It is valid to write `a = b = c`, which means assign `c` to `b` and then assign `b` to `a`. This style is not used in this book.

Some of these operators are beyond the scope of an introductory text and are marked with an asterisk (*) on the right.

(table B-1)

Precedence of Java operators

Operator	Syntax
Postfix operators	
Array access	*«arrayName»*`[`*«index»*`]`
Member access	*«object_or_class»*`.`*«memberName»*
Parameter evaluation	*«methodName»*`(`*«parameterList»*`)`
Postfix increment	*«variable»*`++`
Postfix decrement	*«variable»*`--`
Unary operators	
Prefix increment	`++`*«variable»*
Prefix decrement	`--`*«variable»*
Unary plus	`+`*«expr»*

(table B-1) ***continued***

Precedence of Java operators

Operator	Syntax	
Unary minus	`-«expr»`	
Bitwise complement	`~«expr»`	*
Logical negation	`!«expr»`	
Creation and cast		
Object creation	`new «className»`	
Cast	`(«type»)«expr»`	
Multiplicative operators		
Multiplication	`«expr» * «expr»`	
Division	`«expr» / «expr»`	
Remainder	`«expr» % «expr»`	
Additive operators		
Addition	`«expr» + «expr»`	
Subtraction	`«expr» - «expr»`	
Bit shift operators		
Left shift (propagate sign)	`«expr» << «expr»`	*
Right shift (propagate sign)	`«expr» >> «expr»`	*
Right shift (propagate zero)	`«expr» >>> «expr»`	*
Relational operators		
Less than	`«expr» < «expr»`	
Less than or equal to	`«expr» <= «expr»`	
Greater than	`«expr» > «expr»`	
Greater than or equal to	`«expr» >= «expr»`	
Class membership	`«object» instanceof «className»`	
Equality operators		
Equals	`«expr» == «expr»`	
Not equals	`«expr» != «expr»`	
Bitwise AND operator	`«expr» & «expr»`	*
Bitwise exclusive OR operator	`«expr» ^ «expr»`	*
Bitwise inclusive OR operator	`«expr» | «expr»`	*
Logical AND operator	`«expr» && «expr»`	

(table B-1) *continued*

Precedence of Java operators

Operator	Syntax	
Logical OR operator	*«expr»* `||` *«expr»*	
Conditional	*«expr»* `?` *«expr»* `:` *«expr»*	*
Assignment		
	«var» `=` *«expr»*	
	«var» `+=` *«expr»*	
	«var» `-=` *«expr»*	
	«var» `*=` *«expr»*	
	«var» `/=` *«expr»*	
	«var» `%=` *«expr»*	
	«var» `>>=` *«expr»*	*
	«var» `<<=` *«expr»*	*
	«var» `>>>=` *«expr»*	*
	«var» `&=` *«expr»*	*
	«var» `^=` *«expr»*	*
	«var» `|=` *«expr»*	*