Appendix C Variable Initialization Rules

Variables always have a value. Initialization guarantees that a variable has a known starting value. Without a known starting value, it is hard to have confidence in the correctness of any computations done with the variable. This brief appendix outlines the rules governing the initialization of instance variables, temporary variables, and parameter variables.

Instance and Class Variables

Instance and class variables are always given an initial value, either explicitly by the programmer or implicitly by the compiler. Implicit initializations by the compiler depend on the variable's type, as shown in Table C-1.

(table C-1) *Implicit variable initialization values*

Type	Implicit Initial Value
`byte`, `short`, `int`, `long`	`0`
`boolean`	`false`
`char`	`'\0000'`
`float`	`+0.0f`
`double`	`+0.0`
object reference (including `String`)	`null`

Temporary Variables

Temporary variables are *not* given an initial value by the compiler. The compiler attempts to verify that each temporary variable is initialized before it is used. If the compiler is unable to verify this property, it will issue a compile-time error.

Parameter Variables

Parameter variables are initialized by the corresponding actual parameter in the method's call.

Arrays

Each element of a newly created array is given a default value. The default value depends on the array's type, as shown in Table C-1.