

naming a value, then combining values into new ones

Declaring and initialising a variable *

- a **declaration** states a variable's type and name, such as `int marks`, and reserves it a slot of the right size
- an **initialisation** gives that slot its first value, either on the same line as the declaration or on a later one
- a local variable declared but never given a value cannot be used in an expression, the compiler rejects it as possibly unassigned, though a field of a class is given a default instead
- several variables of the same type may be declared together on one line, their names separated by commas

Naming rules and conventions

- a name must start with a letter, an underscore or a dollar sign, never a digit, and the rest may add digits too
- Java is case sensitive, so `total` and `Total` name two different variables
- a small fixed set of words is reserved by the language itself, such as `class`, `if` and `int`, and none of these may be used as a name
- convention, not a rule the compiler enforces: variable and method names start lower case and grow a capital at each new word, such as `totalMarks`

Local scope against instance scope, the short version

Local variable

- declared inside a method or a block
- exists only while that method or block is running, then it is gone

Instance variable

- declared inside a class, outside any method
- exists for as long as the object holding it exists

the full rules for where a name is visible belong to Statements and Scope, a sibling chapter