

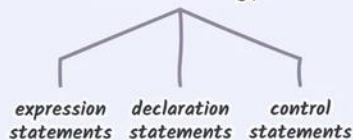
# Statements and Scope

what a line of Java actually is, and where a name it declares can be seen

## Three kinds of statement

- an **expression statement** is an expression turned into a full statement by a semicolon, such as a method call or an assignment on its own line
- a **declaration statement** introduces a new variable, giving it a type and a name, optionally with a first value
- a **control statement** decides which other statements run, or how many times, such as if-else, switch, and every loop
- every one of these ends with a semicolon except the ones that open a block of their own, which end with a curly brace instead

### Statement types



the three families every single Java statement falls into

## A block is a group of statements in one pair of curly braces

\*

- a **block** is any run of statements between a pair of curly braces, treated by Java as a single unit wherever one statement is expected
- a method body is a block, and so is the body that follows an if, an else, or a loop
- a block may be written even where nothing requires one, purely to group statements or to open a fresh scope
- one statement after if or else never needs braces at all, but a second statement written there without them is not controlled by the if, it simply runs every time

## A block's scope against a method's scope

### Method scope

- a local variable declared anywhere directly in the method body is visible for the rest of that method
- it is destroyed the moment the method returns

### Block scope

- a variable declared inside an inner pair of curly braces is visible only until that pair closes
- it is destroyed the moment control leaves that inner block, even though the method carries on

every block scope sits nested inside the method's own scope, never the other way round