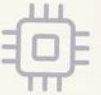
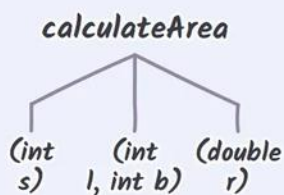


one named block of code, and the special one that builds the object



A method's header, piece by piece

- a method's header lists, in order, the **return type**, the **method name**, and a **parameter list** in parentheses, and the method body follows
- the return type says what kind of value comes back: int, double, String, boolean, char, or void when nothing comes back at all
- the parameter list is empty parentheses when the method takes nothing in, and a comma separated list of type and name pairs when it does
- calling a method means writing its name with an argument for every parameter, in the same order the parameters were declared
- method overloading** is several methods in one class sharing a name, told apart by the number, the types or the order of their parameters



three methods sharing the name `calculateArea`, told apart only by the parameter list each one declares

The four parts of a method header

Return type

int, double, String, boolean, char, or void for nothing returned

Method name

a verb like `calculateArea` or `isEven`, chosen to say what the method does

Parameter list

the inputs the method needs, each written as a type followed by a name

Method body

the statements that run when the method is called, plus its own local variables

void methods against value-returning methods

void method

- declared with `void` in place of a return type
- runs its statements and simply ends, or ends early with a bare `return` statement
- called on its own line, as a statement, since it hands nothing back

value-returning method

- declared with a real type such as `int` or `String` in place of `void`
- every path through the body must reach a `return` statement carrying a value of that type
- called where a value is expected, such as in an assignment or inside another expression

leaving the return type out is never allowed, `void` has to be written on purpose when nothing comes back