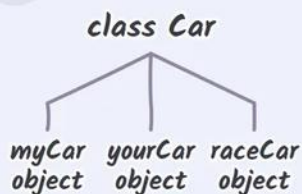


one blueprint, many separate things built from it



A class is a blueprint, an object is a thing built from it

- a **class** is a template that names a group of **fields** (its data) and **methods** (what it can do), written once
- an **object** is one actual thing built from that template while the program runs, also called an instance
- every object made from the same class has the same fields and the same methods, but each object holds its own separate values in those fields
- changing one object's fields never touches another object of the same class, since each has its own copy of the data

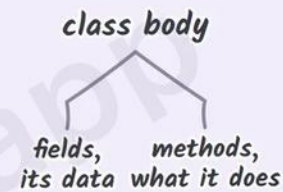


one class, three separate objects built from it, each with its own field values

Fields and methods, the two things a class declares



- **fields** are also called instance variables: each object gets its own copy the moment it is built
- **methods** are the actions an object can perform, usually written to read or change that object's own fields
- a field is declared with a type and a name, the same way any variable is, just written inside the class body rather than inside a method
- a method is written with a return type, a name and a parameter list, and its own body of statements



the two kinds of member a class body declares, nothing else belongs there

What new does, step by step

new Car()
space is set aside

→ then

constructor runs
fields get their first values

→ then

reference returned
a variable can now hold it

A reference variable against a primitive variable

Primitive variable

- holds the actual value itself, such as an int holding 7
- copying it copies the value, and the two variables are independent afterwards

Reference variable

- holds the location of an object, never the object's fields directly
- copying it copies the location, so both variables then point at the same one object

two reference variables can point at the same object, and a change through either one is seen through the other

