

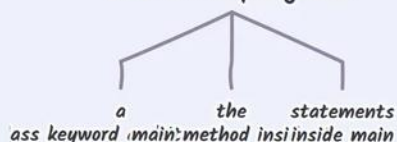
Introduction to Object Oriented Programming using Java

why classes and objects replace one long list of instructions

Procedural code against object oriented code

- procedural programming runs as one long list of instructions, calling functions on data that sits apart from them
- object oriented programming (OOP) bundles data and the functions that act on it into one unit called a class, and every running copy of that class is an object
- Java is written this way from the ground up: even the smallest working program is a class, and every statement lives inside some class
- the shift is from asking what steps run next to asking what objects exist and what each one is responsible for

one Java program



what a Java file is built from, one piece containing the next

The four ideas OOP is built from

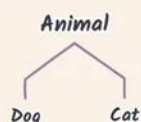
Encapsulation

fields and methods bundled into one class, fields usually kept private so outside code cannot change them directly

Abstraction

showing only what an object can do, hiding how it does it

Inheritance



a new class reusing an existing class's fields and methods instead of copying them

Polymorphism

the same method call behaving differently depending on the object it is called on

From source file to a running program *

MyProgram.java

source code, human readable

→ then

MyProgram.class

bytecode, produced by javac

→ then

JVM

runs the bytecode