

Writing Classes

AP Computer Science A ■ Unit 3

AP Computer Science A

```
class Dog
```

Writing Classes

What we will cover

- 1 Abstraction & program design
- 2 The anatomy of a class
- 3 Constructors & methods
- 4 References & static members
- 5 Scope & the this keyword

Dog

name, age

bark()

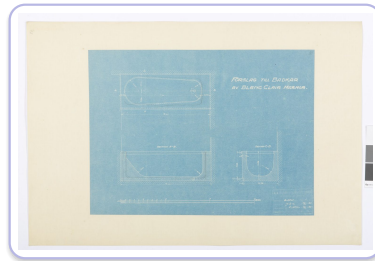
1

Design

Abstraction & program design

Abstraction 抽象 hides detail behind a simple interface, so we manage complexity one piece at a time.

Good **program design** 程序设计 breaks a problem into classes, each with one clear responsibility. A well-designed class is easier to test, reuse, and change without breaking the rest.



Class blueprint

2 Anatomy

The anatomy of a class

class Dog

```
private String name;  
private int age;  
fields 字段 (the object's data)  
public Dog(String n) ... constructor 构造函数  
public void bark() ... methods 方法
```



Jigsaw design

Constructors & methods

Constructor 构造函数

runs once at `new`; sets the object's initial **fields** 字段

Method 方法

a named action; may take **parameters** 形参 and return a value

A method **returns** 返回 a value with `return`; a void method returns nothing.

3

References & static

References & static members

An object variable holds a **reference** 引用 – a pointer to the object, not the object itself.
Passing it lets a method change the same object.

Instance member

belongs to **each object**: `d.bark()`

Static 静态 (class) member

belongs to the **class**:
`Math.sqrt(9)`

Scope & the this keyword

Scope 作用域 is where a name is visible. A local variable lives only inside its method; a field lives as long as the object.

`this` refers to the current object. Use it when a parameter name hides a field:
`this.name = name;` sets the field from the parameter.

Key points

A class has **fields**, a **constructor** & **methods**

The constructor sets initial state at new

Object variables hold a **reference**;
static belongs to the class

`this` names the current object

Worked example – passing a reference

`s` is a `Student` with score 50; we call `tweak(s)`

Java passes the **reference by value**: `tweak` gets a **copy of the arrow**, pointing at the **same** object. So `s.setScore(90)` inside `tweak` **does** change the caller's object – but reassigning `s = new Student()` inside `tweak` **does not**.

You can change what the object **contains**; you cannot change which object the **caller's** variable points to.

Check yourself

- 1 Can a method change the caller's object through a reference?
- 2 Can it change which object the caller's variable points to?
- 3 What does `static` mean – per object, or per class?
- 4 What does `this` refer to?

Answers 1. yes 2. no 3. per class – shared by all objects 4. the current object