Object-oriented programming

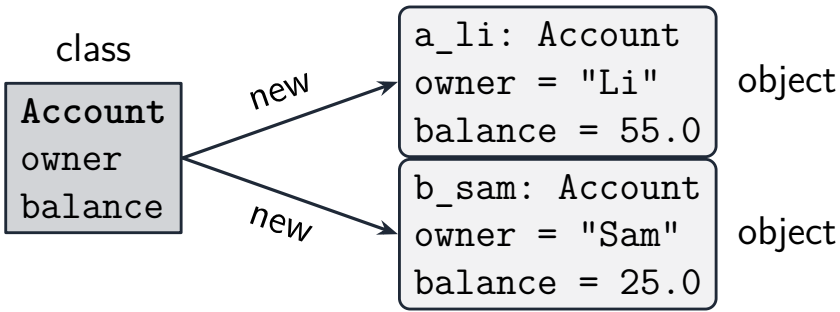
IB DP Computer Science B3 Object-oriented programming full package

Why model the world as objects?

> Write the design once; make one object per student.

- Theme B3 in Python, with Java where its syntax differs.
- Prerequisites: B2 variables, selection, loops and functions.

Class, object and instance



Same attributes, each with its own values.

Writing a class in Python

1. **Make** a_li = Account("Li", 40.0); b_sam = Account("Sam", 20.0)
2. **Call** a_li.deposit(15.0) changes a_li only
3. **Print** a_li.get_balance(), b_sam.get_balance() gives 55.0 20.0

Decision

Each object keeps its own balance.

Two names, one object

After c = a_li, a deposit through c is seen through a_li.

- A variable holds a **reference** to an object.
- Assigning copies the reference, not the object.

Instance and static members

Instance attribute

self.balance: one value per object.

Li and Sam each have their own.

Static (class) attribute

Account.count: one value for the whole class.

Each **constructor** adds 1: after two accounts it is 2.

Encapsulation

Access	Java	Python	Who can use it
private	private	_balance	own methods (Java enforces; Python by convention)
public	public	deposit	any code

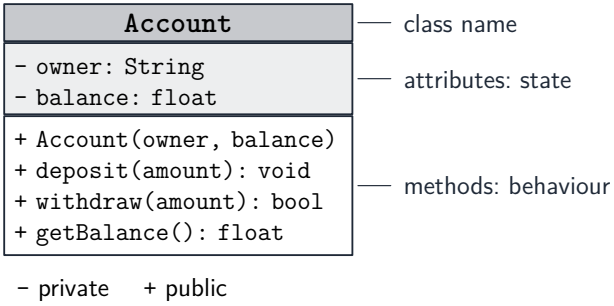
A mutator that checks

1. **Start** acc = Account("Li", 15.0)
2. **withdraw(30.0)** more than the balance: False, balance stays 15.0
3. **withdraw(15.0)** allowed: True, balance becomes 0.0

Decision

A rejected withdrawal leaves the **state** unchanged.

Why encapsulate?



> The class protects its own rules.

Designing a Book class

Attributes (private)	**Methods (public)**
title: String	lend(): bool
isbn: String	give_back(): void
on_loan: bool	is_on_loan(): bool

Advantages and limits of OOP

Advantages

Models real things; easy to discuss.

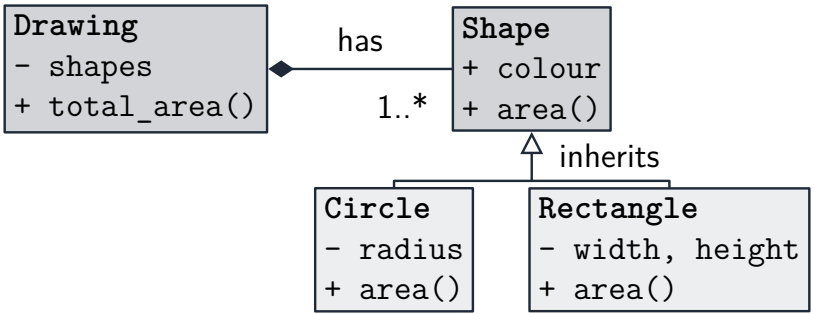
Classes are reused; **encapsulation** protects data.

Disadvantages

More code and planning for a small program.

More memory; a poor design is hard to change.

Inheritance: an is-a link



A circle is a shape.

Polymorphism

1. **List** shapes = [Circle("red", 1.0), Rectangle("blue", 2.0, 3.0)]
2. **Loop** for s in shapes: print(round(s.area(), 2))
3. **Output** 3.14, then 6.0

Decision

One call, different classes, each running its own version.

Abstract classes

> Shape() now raises an error.

- AHL: Python: class Shape(ABC) with @abstractmethod area.
- Java: **abstract class** Shape; abstract double area();

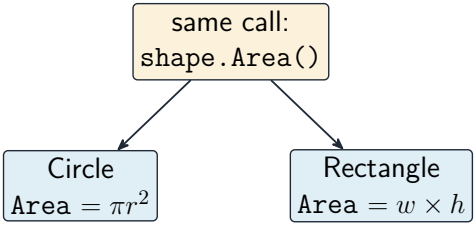
Is a, or has a?

Relationship	Meaning	Example	UML
inheritance	is a	Circle, Shape	hollow triangle
composition	has a; dies with the whole	Drawing, Shapes	filled diamond
aggregation	has a; lives on alone	Team, Players	hollow diamond

Design patterns

Pattern	**Problem**	**Idea**
singleton	exactly one object needed	always return the one **instance**
factory	code should not pick the **subclass**	one **method** decides which to make
observer	others must react to a change	notify every listed observer

One call, different code



same method name — different code runs

> The object decides which code runs

Practice

Your turn

Try these:

A. p = Account("Kai", 30.0); q = p; q.deposit(20.0): p's balance?
B. Why is balance private?
C. How does one loop make a Dog and a Cat speak?
D. Car and Engine; Truck and Vehicle; Club and Member?

Check yourself

