B3 OBJECT-ORIENTED PROGRAMMING · B3.1

Fundamentals of OOP (single class)

Modelling things as objects: **classes** and **objects**, attributes and methods, the **constructor**, static vs instance members, and **encapsulation**.

01 Core building blocks

Class	Blueprint for a type of thing.
Object	An instance of a class.
Attribute	Data / state (e.g. balance).
Method	Behaviour (e.g. deposit()).
Instantiate	Create an object from a class.

02 Members & constructor

Constructor	Runs on creation; sets values.
Instance	Per object (own copy).
Static	Shared by the whole class.
Getter	Public method to read a value.
Setter	Public method to change a value.

03 From blueprint to behaviour

1

Define the class

List its attributes and methods.

2

Instantiate

Create an object; the constructor sets it up.

3

Use methods

Call public methods to read or change state safely.

04 Encapsulation

Bundle	Data + methods in one class.
Info hiding	Attributes are private.
Access	Via public getters/setters.
Why	Control and protect the data.
Analogy	Vending machine buttons.

05 Worked example: BankAccount

interestRate static (shared by all).

__balance private (information hiding).

constructor stores owner + balance.

deposit() rejects amounts ≤ 0.

getBalance() returns the private value.

06 Know the difference

Class vs object	The blueprint versus a specific instance built from it with its own values.	CORE
Static vs instance	One shared copy for the whole class versus a separate copy per object.	MEMBERS
Attribute vs method	The data an object stores versus the actions it can perform.	STRUCTURE
Public vs private	Accessible from outside versus hidden, reachable only via the class's own methods.	ACCESS

07 From a class design to a state trace

The class · private data, public methods

```
class Book:
  __init__(self, title, author, copies):
    self.__title = title
    self.__author = author
    self.__copiesAvailable = copies
  borrowBook(self):
    if self.__copiesAvailable > 0:
      self.__copiesAvailable -= 1
  returnBook(self):
    self.__copiesAvailable += 1
  getCopies(self):
    return self.__copiesAvailable
```

State trace · b = Book("Dune","Herbert",3)

Operation	copiesAvailable	getCopies()
Book(...,3)	3	-
b.borrowBook()	2	-
b.borrowBook()	1	-
b.getCopies()	1	**1**
b.returnBook()	2	-

Why private? Every change goes through a method, so borrowBook() can refuse to lend a copy that does not exist. That control is encapsulation.

Your turn · from copiesAvailable = 2, call borrowBook() three times

Answer: 1, then 0, then the guard blocks the third call, so it stays at **0**.

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Eight things that lose marks if you slip on them. Skim before you walk in.

01

A **class** is the blueprint; an **object** is an instance of it.

02

Attributes = state (data); **methods** = behaviour (actions).

03

The **constructor** runs automatically on instantiation; you don't call it separately.

04

Instance member = per object; **static** member = one copy shared by the class.

05

Encapsulation bundles data with its methods; **information hiding** makes attributes private.

06

Private attributes are reached via **getters/setters**, which can validate changes.

07

"Why private?" The answer is **control and safety**, not just neatness.

08

"Create an object" means **instantiate** the class, not define a new one.