

Object-oriented programming · supported focus

International Baccalaureate · IB Diploma · Computer Science · SL

Scope and prerequisites

Supported SL focus. First assessment 2027 target; official PDF returns 403; older acquired brief is final assessment 2026. Remaining guide, assessment and practical requirements retain their recorded holds.

Prerequisites: read the stated quantities and units, use arithmetic and the model conditions below. Each lesson develops its own method before independent transfer.

These are original or explicitly fictional teaching examples, not actual measurements or completed assessed learner investigations.

Objects: state, behaviour and an interface

■ What would explain this observation?

- Two bank-account model objects can respond to the same deposit operation while holding different balances.
- Start with a prediction. State the quantities or features you would compare, then decide what evidence could distinguish two explanations.

■ Build the model

- A class defines a type with state and behaviour. An object is an **instance**. **Encapsulation** controls access through an interface so operations can preserve invariants. Inheritance models an appropriate type relationship; composition models an object containing or using another object.
- encapsulation: Controlling access to state through an interface; instance: An individual object of a class.



■ Choose evidence that can test it

- A method call acts on a particular instance. State changes should satisfy preconditions and postconditions. Polymorphism lets code use a common interface with different implementations when the contract is respected.
- Implement a small synthetic account or inventory model. Test two independent instances, rejected invalid operations and boundary values. Keep the model away from real financial accounts and credentials.

■ Work from known quantities

- State the known values and their units. Choose the relation because its assumptions fit this case, then rearrange before substitution.
- Known: object A starts at 40 units and receives 15; object B starts at 20 and receives 5. Final balances are 55 and 25. Their total is 80, but neither individual object balance is 80.

Example:

An instance starts at 12 units, adds 8 and removes 3. Find its final value. Use the same sequence: known quantities → model → relation → substitution → unit and interpretation.

■ Check the conclusion and its limits

- A class is not the same thing as one instance. Inheritance is not automatically better than composition. Merely hiding a field does not prove that all operations preserve valid state.
- Return to the original observation. Explain what the result supports, which conditions it assumes, and one way to test a competing explanation.

Warn:

Two instances of a class must always share one balance. This claim is false: A class is not the same thing as one instance. Inheritance is not automatically better than composition. Merely hiding a field does not prove that all operations preserve valid state.

Key:

Objects: state, behaviour and an interface: A method call acts on a particular instance. State changes should satisfy preconditions and postconditions. Polymorphism lets code use a common interface with different implementations when the contract is respected.

■ Runnable trace and boundary

```
class Account:
    def __init__(self, balance):
        self.balance = balance
    def withdraw(self, amount):
        if amount < 0 or amount > self.balance:
            return False
        self.balance -= amount
        return True

a, b = Account(15), Account(20)
print(a.withdraw(30), a.balance, b.balance)
print(a.withdraw(15), a.balance, b.balance)
```

Expected output:

```
False 15 20
True 0 20
```

Rejection returns before mutation. With nonnegative initial balances, validated withdrawals preserve $\text{balance} \geq 0$. This small teaching class exposes attributes; a production design must control all mutation paths, not assume the method alone enforces encapsulation.