

Exercise walk-through

7.1

Objects: state, behaviour and an interface

International Baccalaureate · IB Diploma · Computer Science · SL

7.1 Objects: state, behaviour and an interface

Name: _____ Class: _____ Date: _____ **Total: 10 marks**

KEY WORDS instance • encapsulation

Objective

Objects: state, behaviour and an interface. Supported SL concept preparation; original classroom practice, not an official IB paper or assessed learner work.

1 Key knowledge and worked examples

■ 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 individ-

ual 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.

Exercises 2.1-2.3

2.1 What should prevent an invalid state change? Choose one answer.

- **A** An operation that checks and preserves the invariant
- **B** Making every field publicly writable
- **C** Assuming no user makes an error

[1]

2.2 An instance starts at 12 units, adds 8 and removes 3. Find its final value. Show the relation, substitution and units. [2]

2.3 Decide whether this claim is true and justify: Two instances of a class must always share one balance. [2]

Exercises 2.1-2.3

2.1 What should prevent an invalid state change? Choose one answer.

- **A** An operation that checks and preserves the invariant
- **B** Making every field publicly writable
- **C** Assuming no user makes an error

[1]

2.1 A 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.

2.2 An instance starts at 12 units, adds 8 and removes 3. Find its final value. Show the relation, substitution and units.

[2]

2.2 final instance state= $12+8-3=17$ units.

2.3 Decide whether this claim is true and justify: Two instances of a class must always share one balance.

[2]

2.3 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.

Exercises 2.4·3.1

2.4 Controlling access to state through an interface: write the technical term. [1]

3.1 Independent accounts start at 10 and 20. Credit 5 to the first, then try to withdraw 30 under a no-overdraft rule. Give final states and the invariant. [4]

Exercises 2.4·3.1

2.4 Controlling access to state through an interface: write the technical term. [1]

2.4 encapsulation. encapsulation means Controlling access to state through an interface.

3.1 Independent accounts start at 10 and 20. Credit 5 to the first, then try to withdraw 30 under a no-overdraft rule. Give final states and the invariant. [4]

3.1 States are 15 and 20. Rejected withdrawal leaves state unchanged. Instance attributes are independent; validated methods preserve $\text{balance} \geq 0$.