Exercise walk-through

Programming Paradigms ▪ 20.1

eXercise

You must be able to

- explain what a **paradigm** is and the four kinds
- describe the **OOP** features (encapsulation, inheritance, polymorphism, abstraction)
- give examples of **declarative** (functional, logic, SQL) programming

Method — The four paradigms

A **paradigm** is a style of structuring programs.

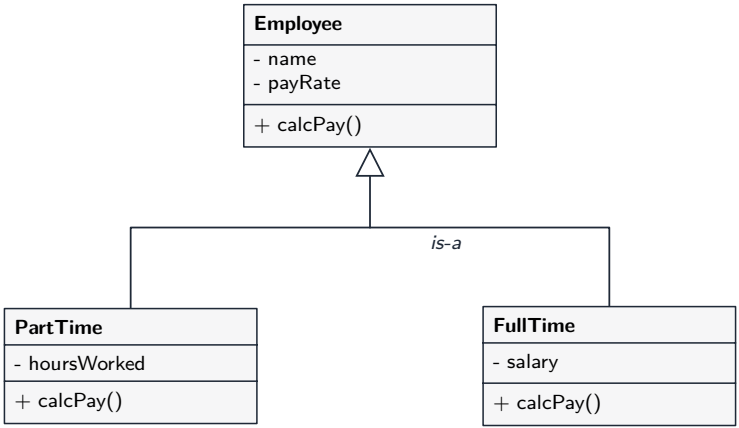
- **Low-level** — machine/assembly code, close to the hardware (fast, but architecture-specific).
- **Imperative (procedural)** — a **sequence of commands** that change state (C, Python).
- **Object-oriented (OOP)** — programs built from **objects** (data + methods).
- **Declarative** — say **what**, not **how** (SQL, Prolog, Haskell).

Method — OOP features

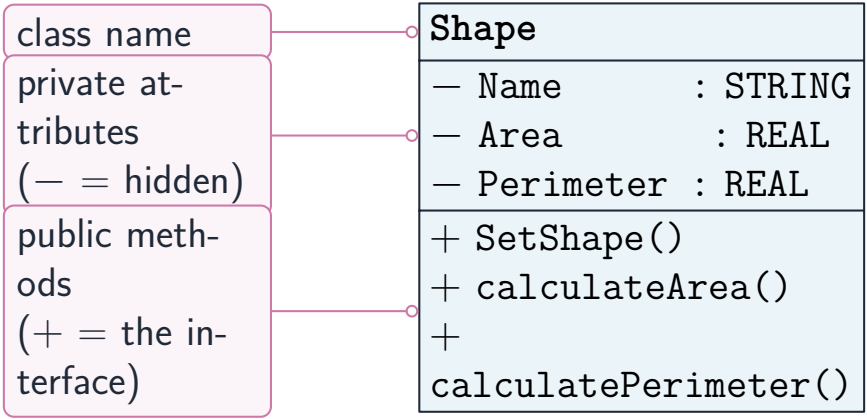
An **object** is an **instance** of a **class** (data = **attributes**, operations = **methods**). The pillars:

- **encapsulation** — data is **hidden** behind methods (outside code uses public methods only).
- **inheritance** — a **subclass** specialises a **superclass** (an "is-a" relationship):
- **polymorphism** — the **same method call** behaves differently per object type (each `Shape` has its own `Area()`).
- **abstraction** — show a simple interface, hide the implementation.

Method — OOP features



Method — OOP features



A class diagram models an object-oriented design

Method — Writing a class (pseudocode)

A class declares **private** attributes and **public** methods, including a **constructor** (`NEW`) that sets up a new object; a subclass uses **INHERITS**:

```
CLASS Employee
    PRIVATE Name : STRING
    PRIVATE PayRate : REAL

    PUBLIC PROCEDURE NEW(GivenName : STRING, GivenRate : REAL)
        Name ← GivenName
        PayRate ← GivenRate
    ENDPROCEDURE

    PUBLIC FUNCTION GetName() RETURNS STRING
        RETURN Name      // a "get" method reads a private attribute
    ENDFUNCTION
ENDCLASS

CLASS Manager INHERITS Employee
    PRIVATE Bonus : REAL
ENDCLASS
```

Method — Declarative

Functional uses **pure functions** (no side effects); **logic** states facts/rules (Prolog); **SQL** queries data — `SELECT * FROM Customer WHERE Country = 'UK'` says **what**, not **how**.

Practice 2.1 ▪ 1 mark

State what a **programming paradigm** is.

Solution 2.1

Worked steps

A **style/approach to structuring programs**, with its own concepts and language features B1.

Practice 2.2 ▪ 2 marks

Name the **four** features (pillars) of **OOP**.

Solution 2.2

Method: OOP features

Worked steps

Encapsulation, inheritance, polymorphism, abstraction B1 B1 *for two, for all four.*

Practice 2.3 ▪ 1 mark

State what **encapsulation** means.

Solution 2.3

Worked steps

An object's **data is hidden** and accessed **only through its methods** (not directly) B1.

Practice 2.4 ▪ 1 mark

Give **one** example of a **declarative** language.

Solution 2.4

Worked steps

Any one: **SQL**, **Prolog**, **Haskell**, Lisp, F# B1.

Exam-style 3.1 ▪ 3 marks

Describe what is meant by the OOP feature **inheritance**, with an **example**.

Solution 3.1

Worked steps

A **subclass** inherits the **attributes and methods** of a **superclass** B1 and can add or **override** its own B1; example: a `Manager` inherits from `Employee` (“is-a”) B1.

Exam-style 3.2 ▪ 2 marks

Identify the OOP feature that **restricts external access to an object's data**, and state **one** benefit of it.

Solution 3.2

Worked steps

Encapsulation B1; benefit: it **protects the data** from invalid changes / lets the internals change without breaking other code B1.

Exam-style 3.3 ▪ 2 marks

Explain what is meant by **polymorphism**.

Solution 3.3

Worked steps

Different object types respond to the **same method call** in their **own way** B1, so the caller need not know the exact type (e.g. `Circle` and `Rectangle` each implement `Area()`) B1.

Exam-style 3.4 ▪ 2 marks

State **one** characteristic of **low-level** programming and **one** of **declarative** programming.

Solution 3.4

Method: The four paradigms

Worked steps

Low-level: any one — close to hardware, direct register/memory access, fast/compact, architecture-specific B1. Declarative: any one — you state **what** not **how**, no explicit step-by-step control flow B1.

Exam-style 3.5 ▪ 4 marks

A class `Book` has a private attribute `Title` of type `STRING`. Write pseudocode for a **constructor** that sets `Title`, and a **get method** `GetTitle()` that returns it.

Solution 3.5

> Method: Writing a class (pseudocode)

Worked steps

```
PUBLIC PROCEDURE NEW(GivenTitle : STRING)
    Title ← GivenTitle
ENDPROCEDURE

PUBLIC FUNCTION GetTitle() RETURNS STRING
    RETURN Title
ENDFUNCTION
```

M1 A1 M1 A1 *constructor 'NEW' with a parameter, sets 'Title', 'GetTitle()' RETURNS STRING, 'RETURN Title'*