Solutions

Each answer shows the steps, then the **result**.

2.1

- a **style/approach to structuring programs**, with its own concepts and language features

2.2

- **encapsulation, inheritance, polymorphism, abstraction**

2.3

- an object's **data is hidden** and accessed **only through its methods** (not directly)

2.4

- any one: **SQL**, **Prolog**, **Haskell**, Lisp, F#

3.1

- a **subclass** inherits the **attributes and methods** of a **superclass**
- and can add or **override** its own
- example: a `Manager` inherits from `Employee` ("is-a")

3.2

- **encapsulation**
- benefit: it **protects the data** from invalid changes / lets the internals change without breaking other code

3.3

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

3.4

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

3.5

- constructor `NEW` with a parameter; getter returns `Title`

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

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