Solutions

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

2.1

- any one: makes code **self-documenting/clearer**; lets the compiler **reject illegal values**; groups related values that belong together

2.2

- `TYPE … = ( … )` with the seven named values
- `TYPE Day = (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)`

2.3

- the **memory address** of another variable (or `NULL` for no target)

2.4

- to **follow the pointer** and reach (read or change) the **variable it points to**

3.1

- `TYPE … = (…)` with the values, then declare and assign

```
TYPE Vehicle = (M100, M230, T101, T102)
DECLARE MyTaxi : Vehicle
MyTaxi ← T101
```

3.2

(i) `TYPE PNode = ^TNode`

(ii) `NEW` then dereference and assign

```
p ← NEW TNode
p^.Value ← 5
```

3.3

- a **set** stores **unordered, unique** values and supports a **membership test** — exactly what "is this colour available?" needs
- test: `IF Green IN Available THEN …`

3.4

- a **class** is the **definition/template** (attributes + methods)
- an **object** is an **instance** of that class, holding its own data

3.5

(a)

```
TYPE Choice
    DECLARE MemberName : STRING
    DECLARE ActivityName : STRING
    DECLARE ChoiceDate : DATE
ENDTYPE
```

- `TYPE`/`ENDTYPE`; all three fields; correct types; correct syntax

(b) `DECLARE ThisChoice : Choice`

(c) `ThisChoice.MemberName ← "Ahmed"`

- `ThisChoice.ActivityName ← "Tennis"`
- `ThisChoice.ChoiceDate ← 12/03/2027`

(d)

```
TYPE Activity
    DECLARE ActivityName : STRING
    DECLARE Members : ARRAY[1:20] OF STRING
ENDTYPE
```

- `TYPE`/`ENDTYPE`; the array of `STRING`; bounds `1:20`

(e) `DECLARE Paid : BOOLEAN` inside the type

- `BOOLEAN`, because the value can only be paid or not paid — two states, and no other value is valid

(f) the three values belong to **one** booking, so keeping them in one record means they can be passed to a procedure, stored in an array or written to a file as a single item

- they cannot drift out of step with each other the way three separate variables can