```
┌──────────────────────── MODULE Door ────────────────────────┐
EXTENDS TLC, Naturals

CONSTANTS Closed, Opening, Open, Closing

VARIABLES state, pc

vars ≜ ⟨state, pc⟩

ProcSet ≜ {"CmdOpen"} ∪ {"CmdClose"} ∪ {"FinishOpening"} ∪ {"FinishClosing"}

Init ≜  Global variables
        ∧ state = Closed
         Process CmdOpen
        ∧ pc = [self ∈ ProcSet ↦ CASE self = "CmdOpen" → "OpenDoor"
                                    □  self = "CmdClose" → "CloseDoor"
                                    □  self = "FinishOpening" → "CompleteOpen"
                                    □  self = "FinishClosing"   → "CompleteClose"]

OpenDoor ≜ ∧ pc["CmdOpen"] = "OpenDoor"
           ∧ state = Closed
           ∧ state′ = Opening
           ∧ pc′ = [pc EXCEPT ![“CmdOpen”] = "Done"]

CmdOpen ≜ OpenDoor

CloseDoor ≜ ∧ pc["CmdClose"] = "CloseDoor"
            ∧ state = Open
            ∧ state′ = Closing
            ∧ pc′ = [pc EXCEPT !["CmdClose"] = "Done"]

CmdClose ≜ CloseDoor

CompleteOpen ≜ ∧ pc["FinishOpening"] = "CompleteOpen"
               ∧ state = Opening
               ∧ state′ = Open
               ∧ pc′ = [pc EXCEPT !["FinishOpening"] = "Done"]

FinishOpening ≜ CompleteOpen

CompleteClose ≜ ∧ pc["FinishClosing"] = "CompleteClose"
                ∧ state = Closing
                ∧ state′ = Closed
                ∧ pc′ = [pc EXCEPT !["FinishClosing"] = "Done"]

FinishClosing  ≜ CompleteClose

Next ≜ CmdOpen ∨ CmdClose ∨ FinishOpening ∨ FinishClosing
           ∨ Disjunct to prevent deadlock on termination
```

```
                    ((∀ self ∈ ProcSet : pc[self] = "Done") ∧ UNCHANGED vars)

Spec ≜ Init ∧ □[Next]_vars

Type invariant definition (repeated from PlusCal for clarity)
TypeOK ≜ state ∈ {Closed, Opening, Open, Closing}

Optional: Define a property, e.g., the door never gets stuck opening forever.
This requires fairness. For simplicity, we'll focus on the TypeOK invariant check.
StuckOpening ≜ state = Opening ⇒ ◇ (state = Open)

END TRANSLATION
```