# HIDDEN LOGICAL DEPENDENCY

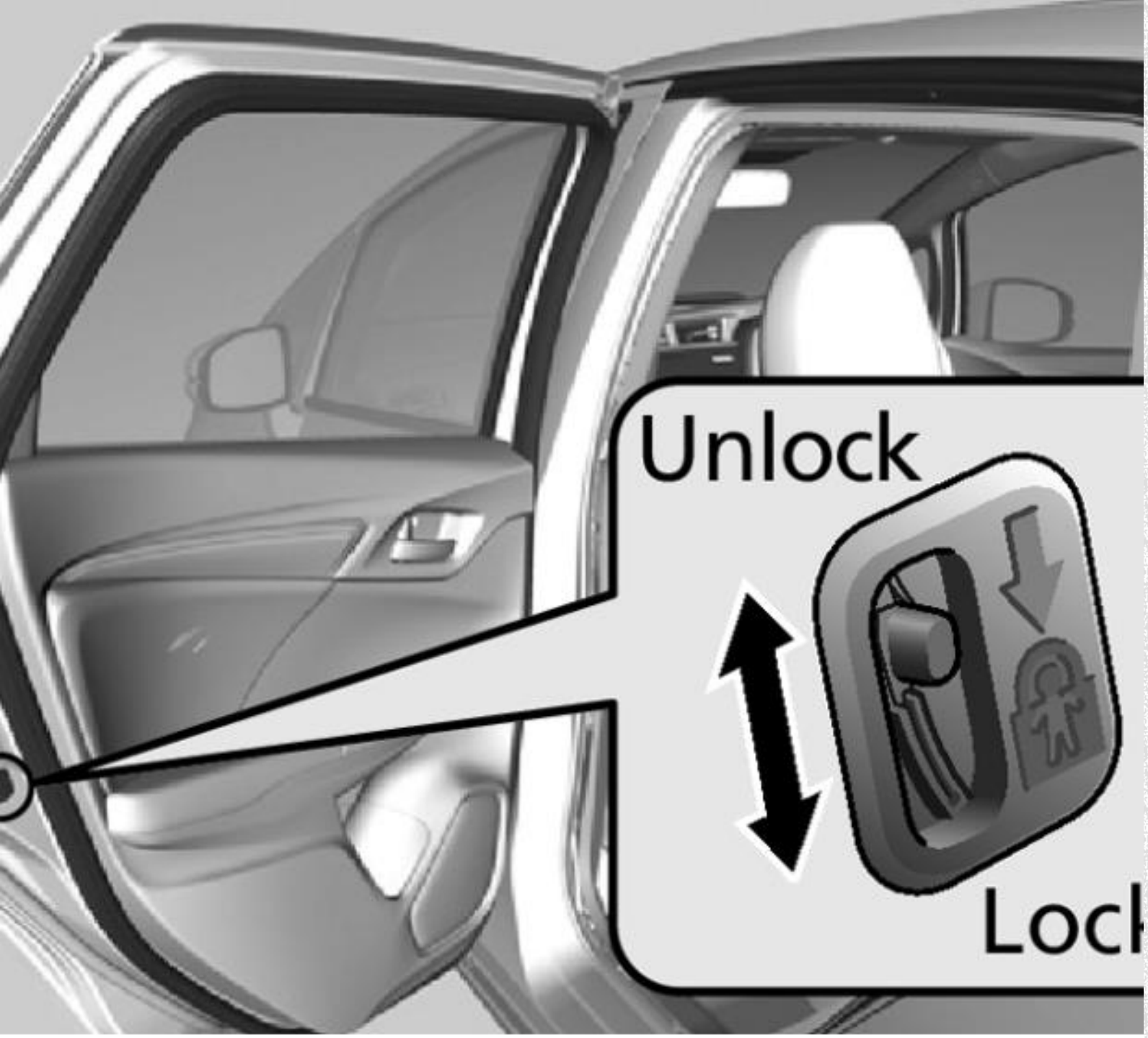
Ferose Khan J

# WHAT IS IT?

It refer to situations where the behavior or outcome of a piece of code depends on conditions or inputs that are not readily apparent or explicitly communicated. These dependencies can lead to confusion, unexpected behavior, and difficult-to-debug issues. Understanding and managing hidden logical dependencies is crucial for writing clean, maintainable, and reliable code.

# SITUATIONS WHEN BEHAVIOR OF A PIECE OF CODE DEPENDS ON

- Global variables
- Implicit Order of Operations
- Unspecified Input Assumptions
- External State Dependencies




# Real World Example – Child Lock in Car

- The inner door lock depends on the child lock
- However, when the door is locked, there is no indication about the child lock is visible to the user around the lock handle
- This introduces a hidden dependency and first-time users experience some annoyance

# LET'S LOOK AT THE SAME IN SOFTWARE


- Car has 4 doors
- Door has a lock
- Door can be opened
- Door can be closed


Door shall have a child lock

If child lock is enabled, then door cannot be opened from  inside

# AVOID HIDDEN LOGICAL DEPENDENCIES

- Follow encapsulation and modularity
- Make things explicit in documentation, API design, requirements and communication
- Validate input data and provide meaningful errors
- Avoid side effects and prefer immutable data from the functional programming paradigm
- Write scenario based good unit tests

# THANKS