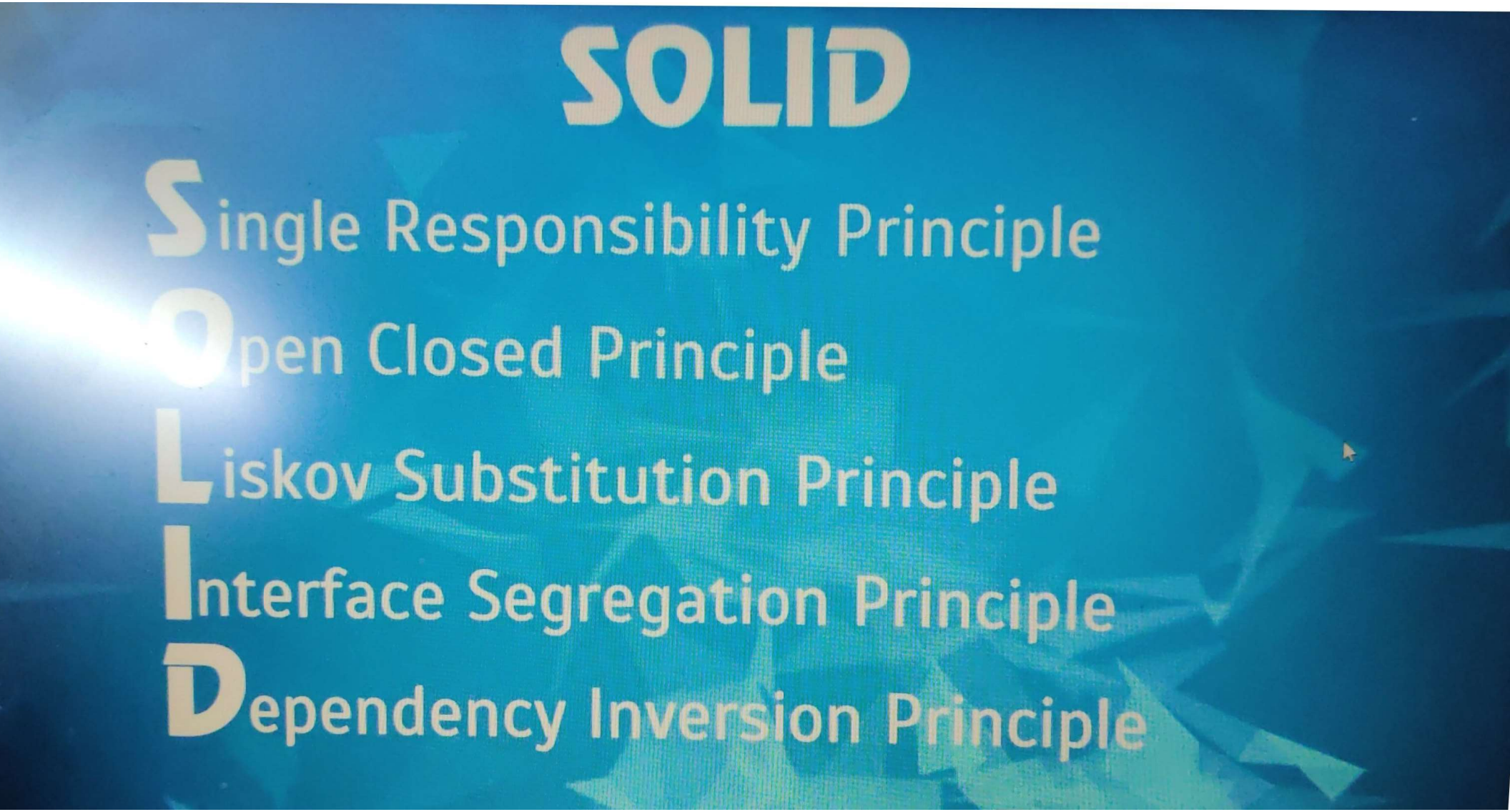
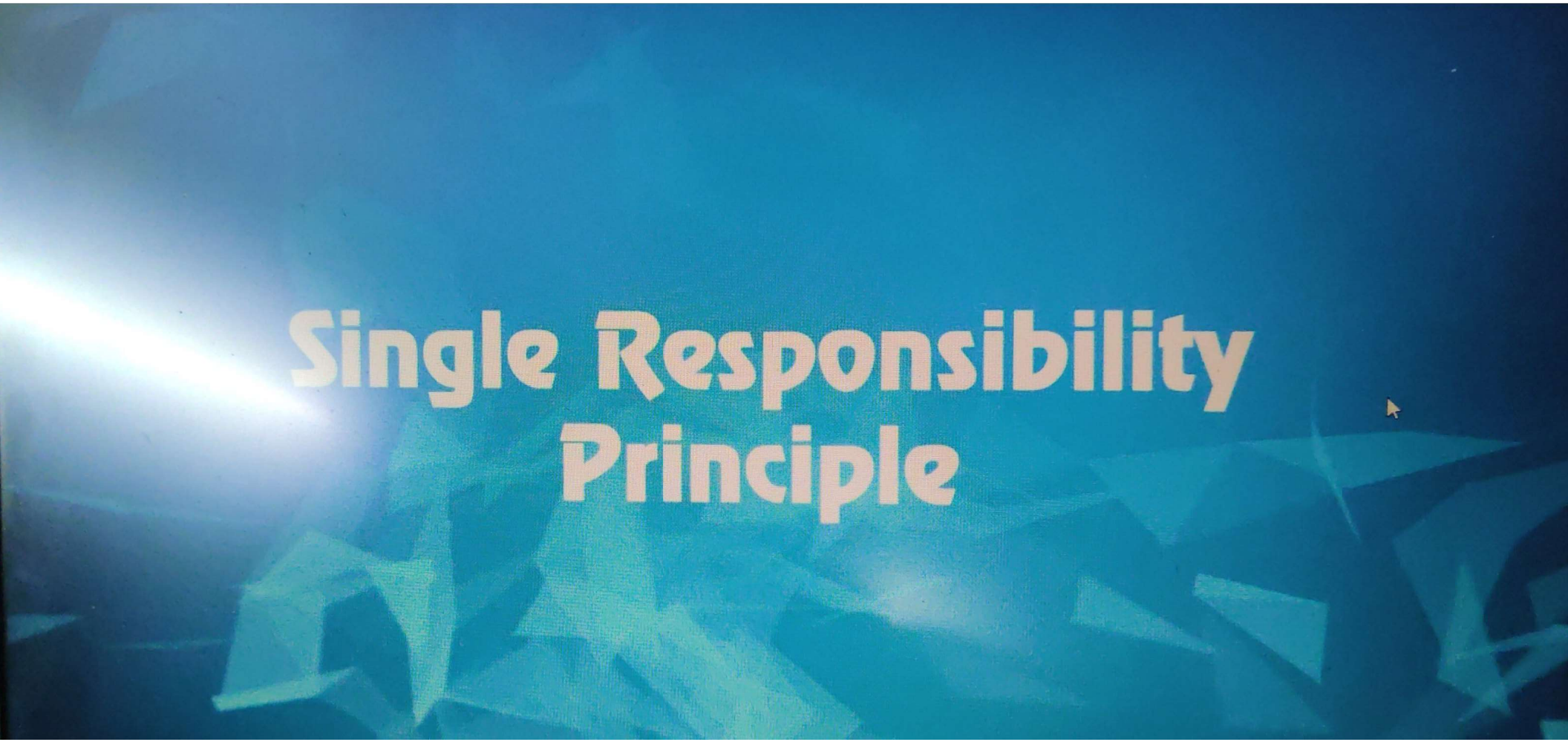
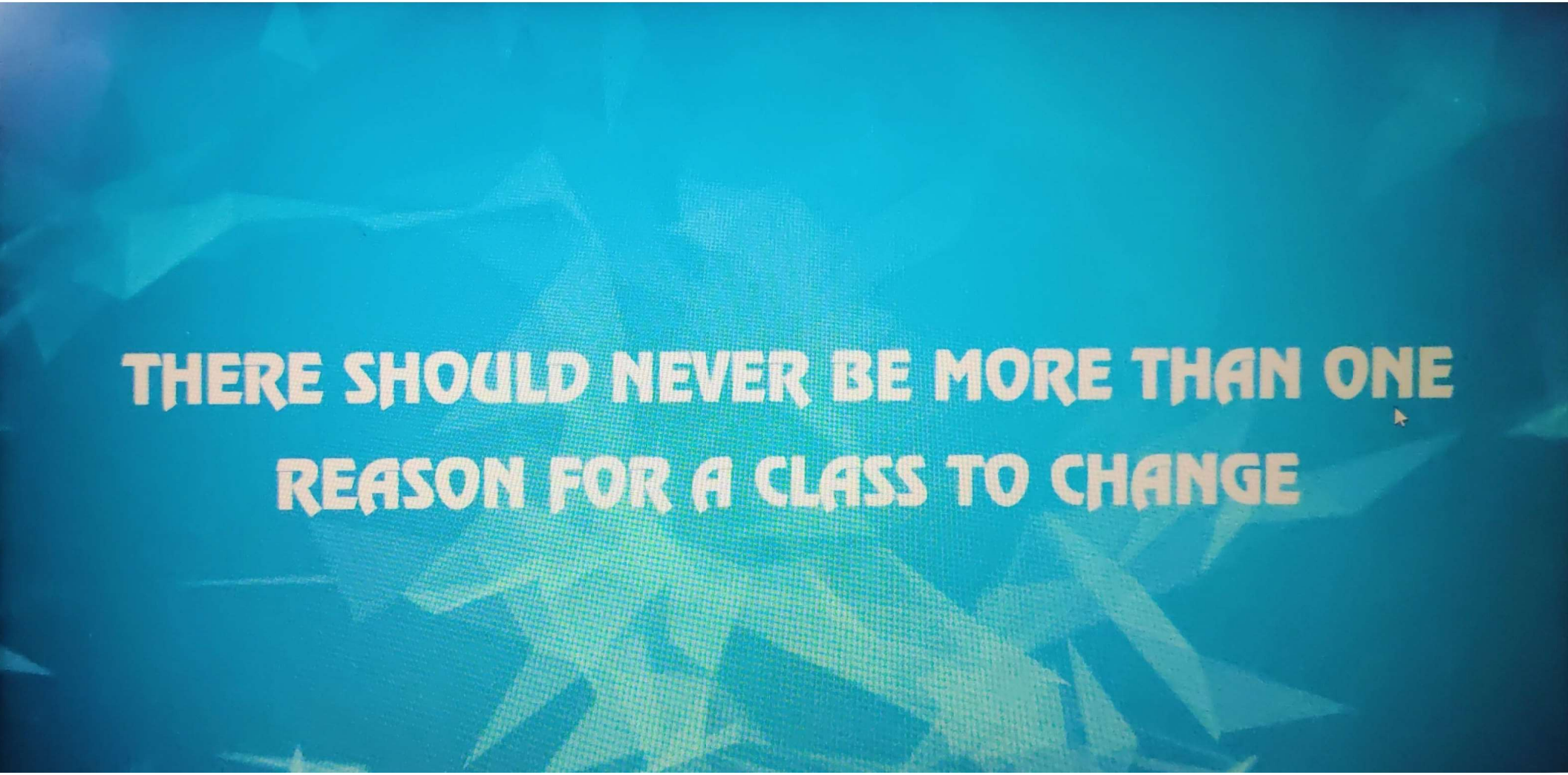
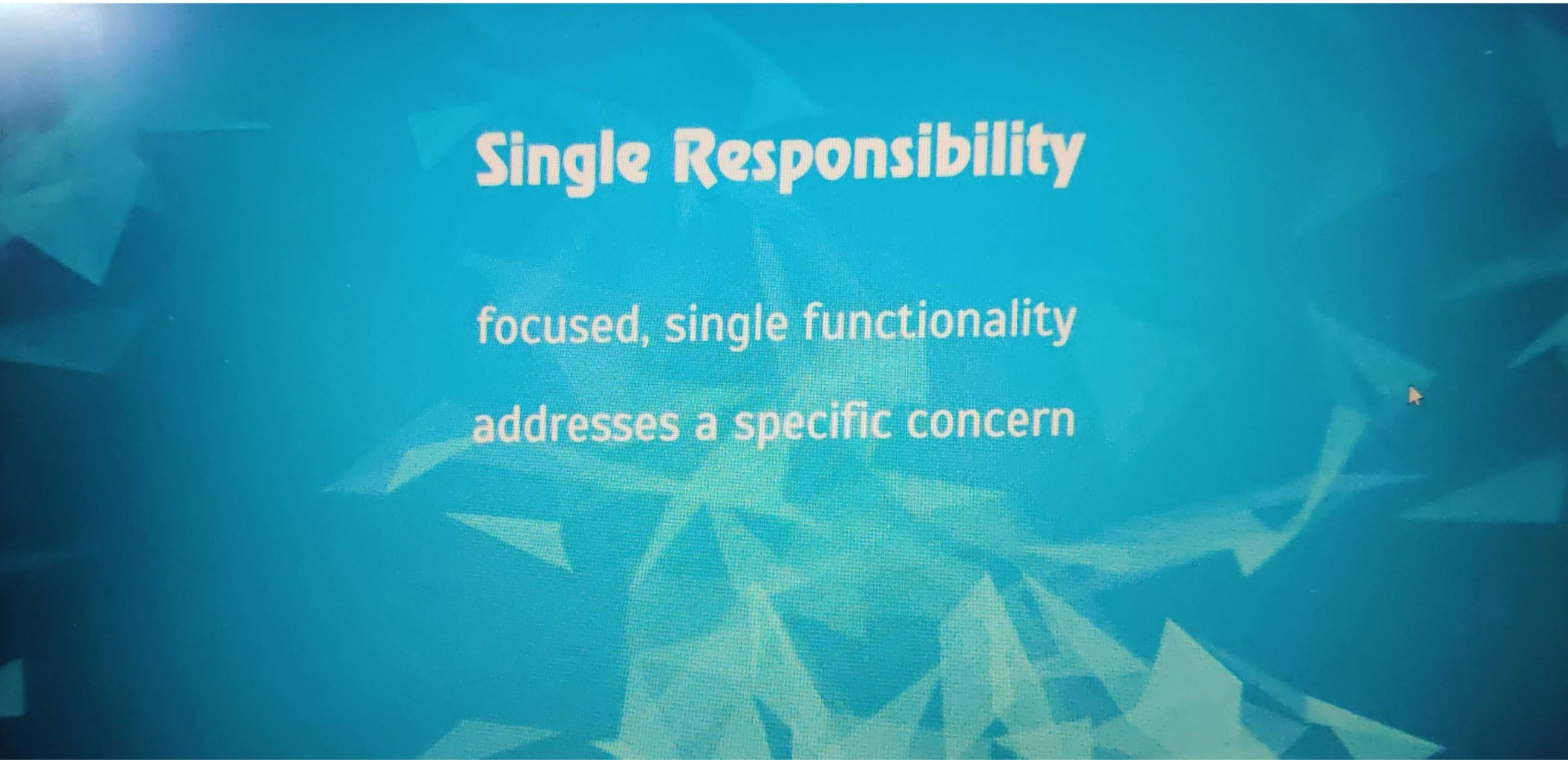
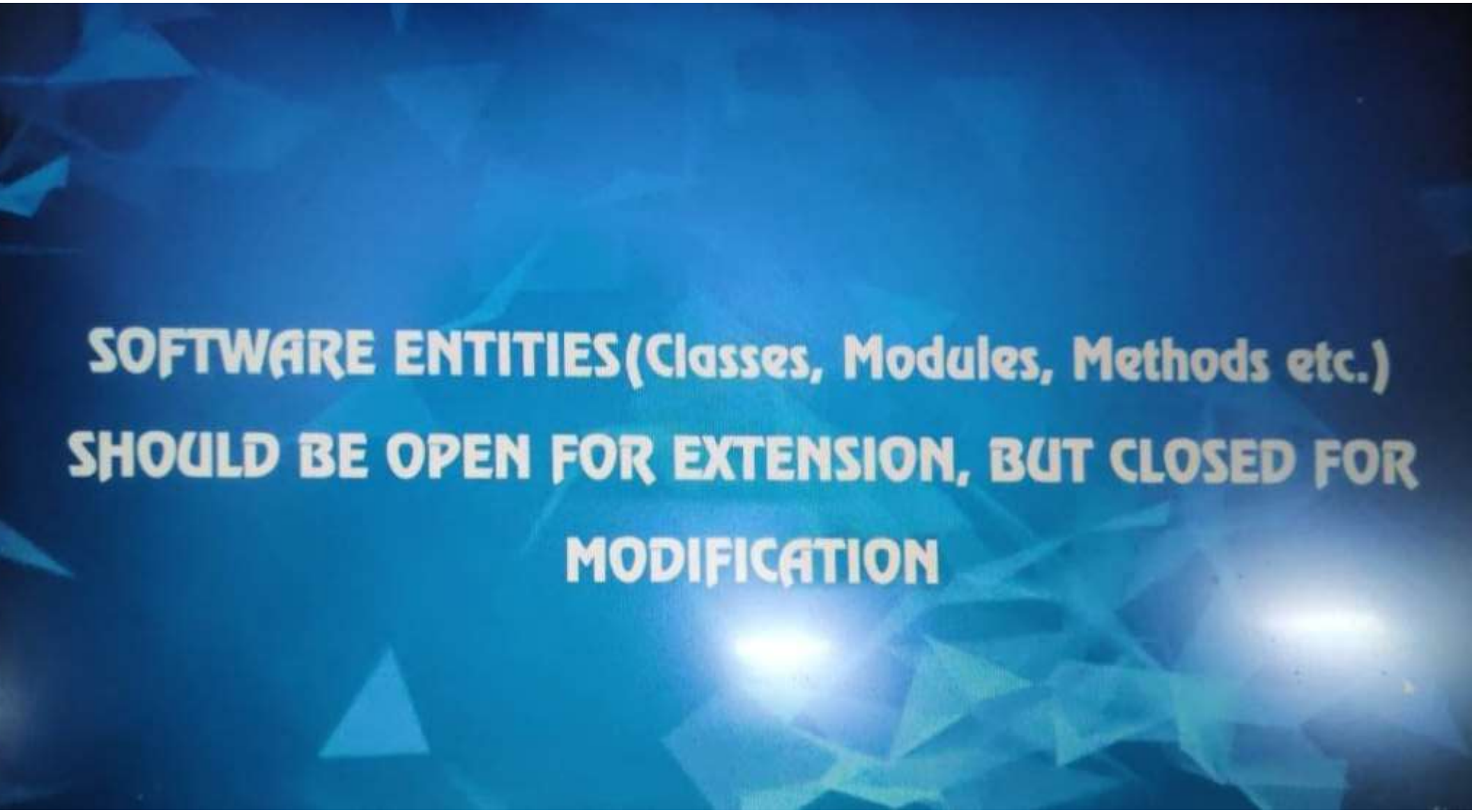

SOLID
Single Responsibility Principle
Open Closed Principle
Liskov Substitution Principle
Interface Segregation Principle
Dependency Inversion Principle


Single Responsibility
Principle

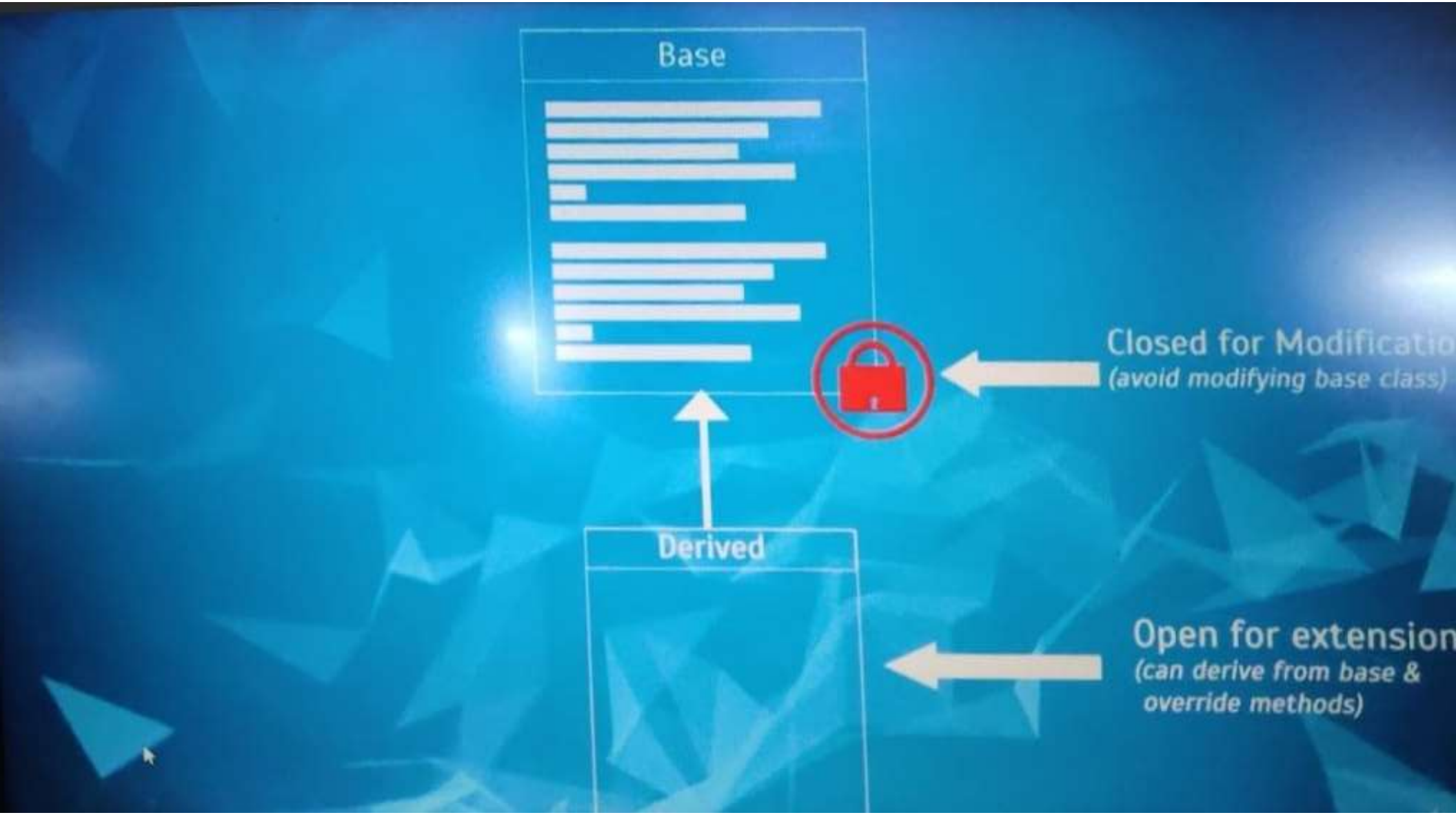

THERE SHOULD NEVER BE MORE THAN ONE
REASON FOR A CLASS TO CHANGE

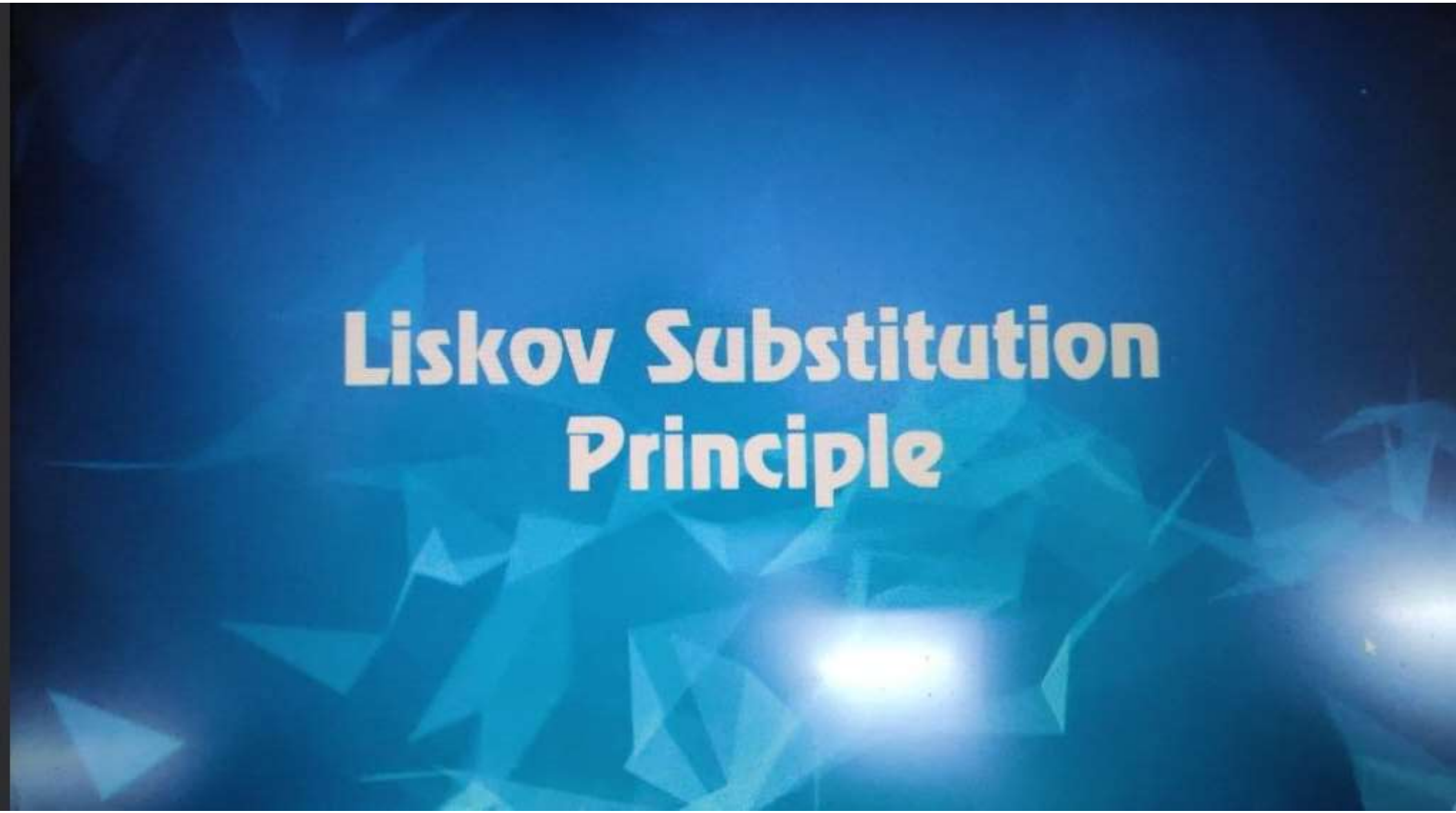

Single Responsibility
focused, single functionality
addresses a specific concern

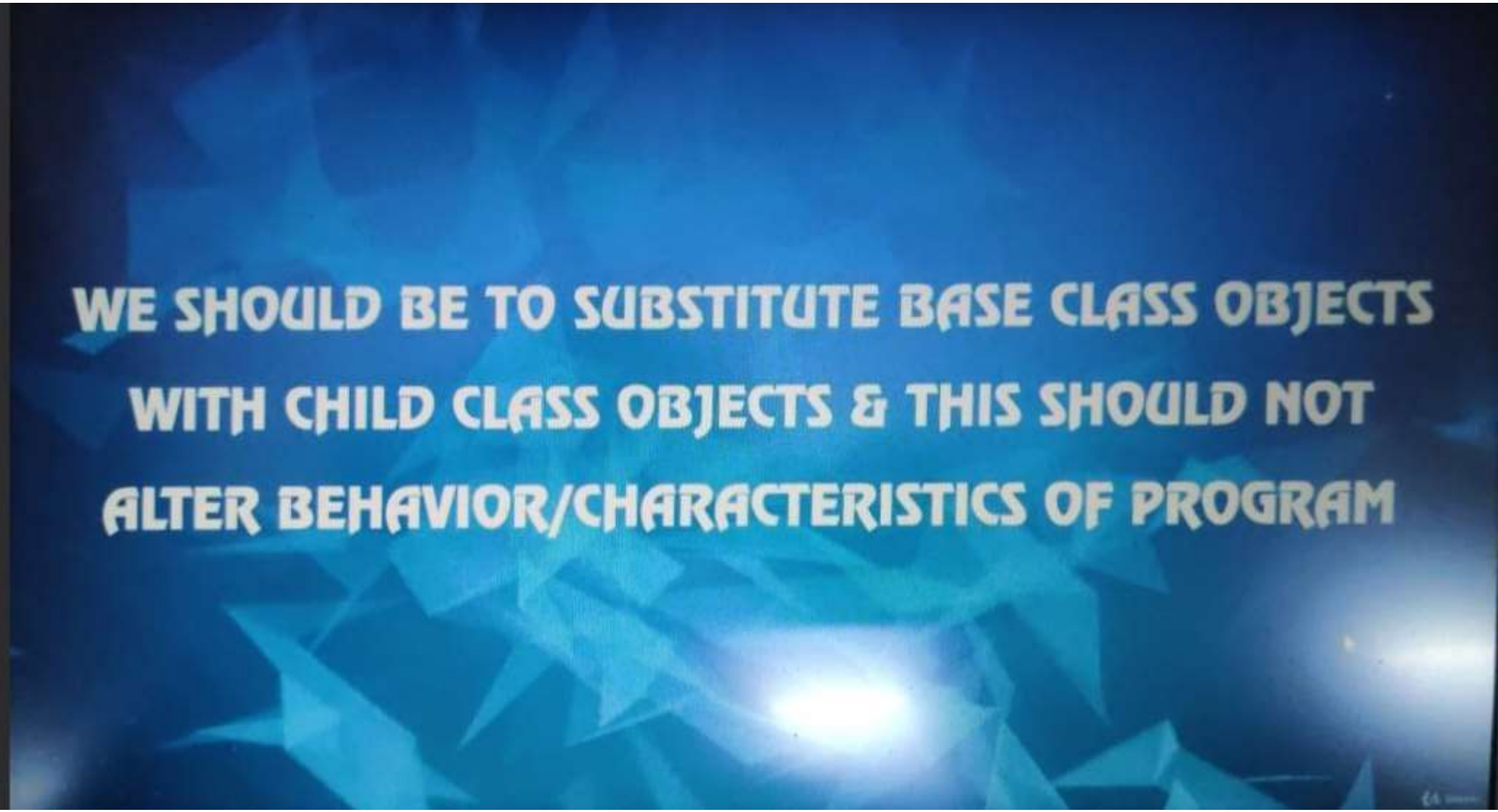

Open - Closed
Principle


SOFTWARE ENTITIES(Classes, Modules, Methods etc.)
SHOULD BE OPEN FOR EXTENSION, BUT CLOSED FOR
MODIFICATION

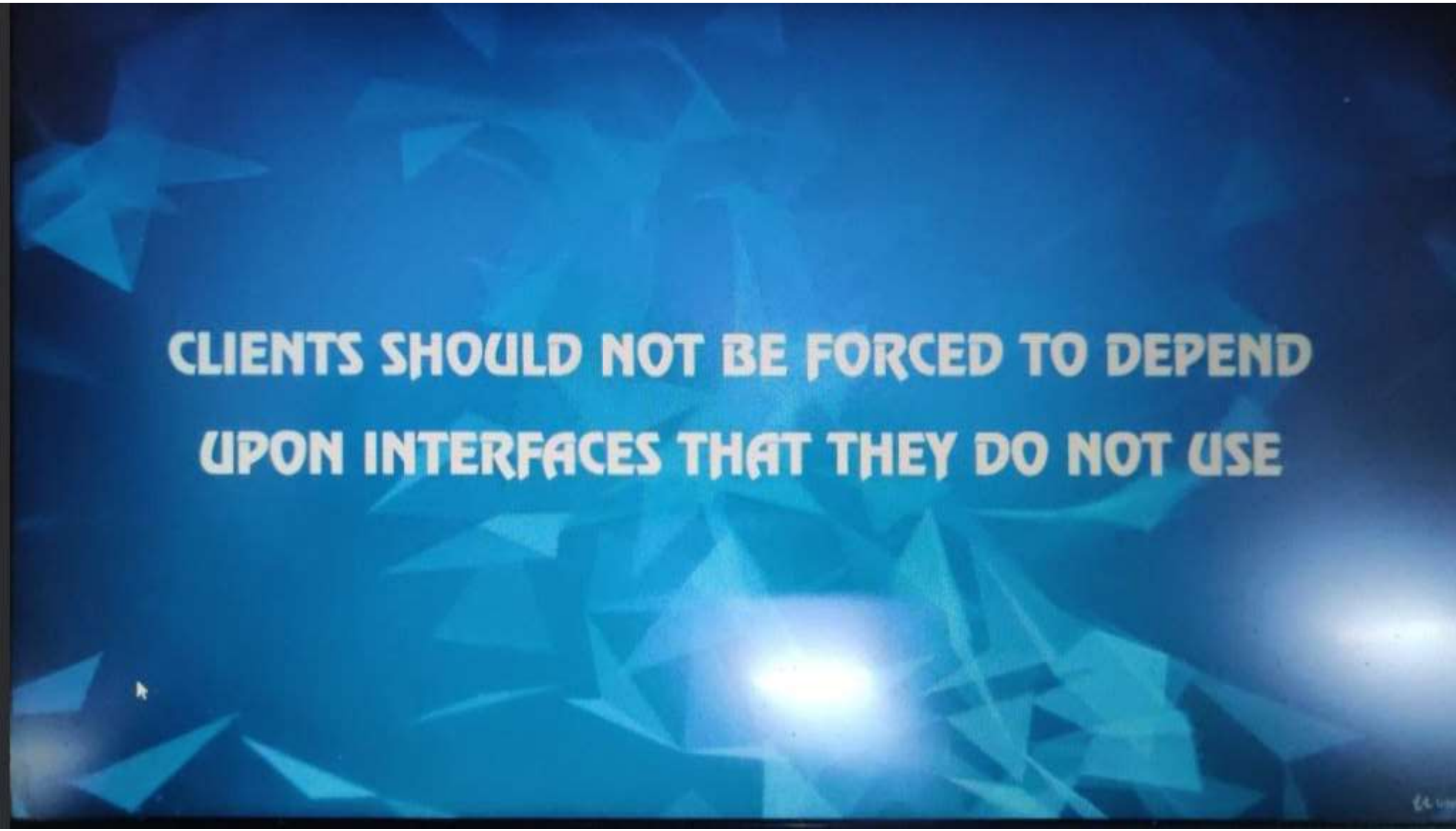

Open for Extension
Extend existing behavior
Closed for Modification
Existing code remains unchange


Base
Closed for Modificatio
(avoid modifying base class)
Derived
Open for extension
(can derive from base &
override methods)


Liskov Substitution
Principle


WE SHOULD BE TO SUBSTITUTE BASE CLASS OBJECTS
WITH CHILD CLASS OBJECTS & THIS SHOULD NOT
ALTER BEHAVIOR/CHARACTERISTICS OF PROGRAM

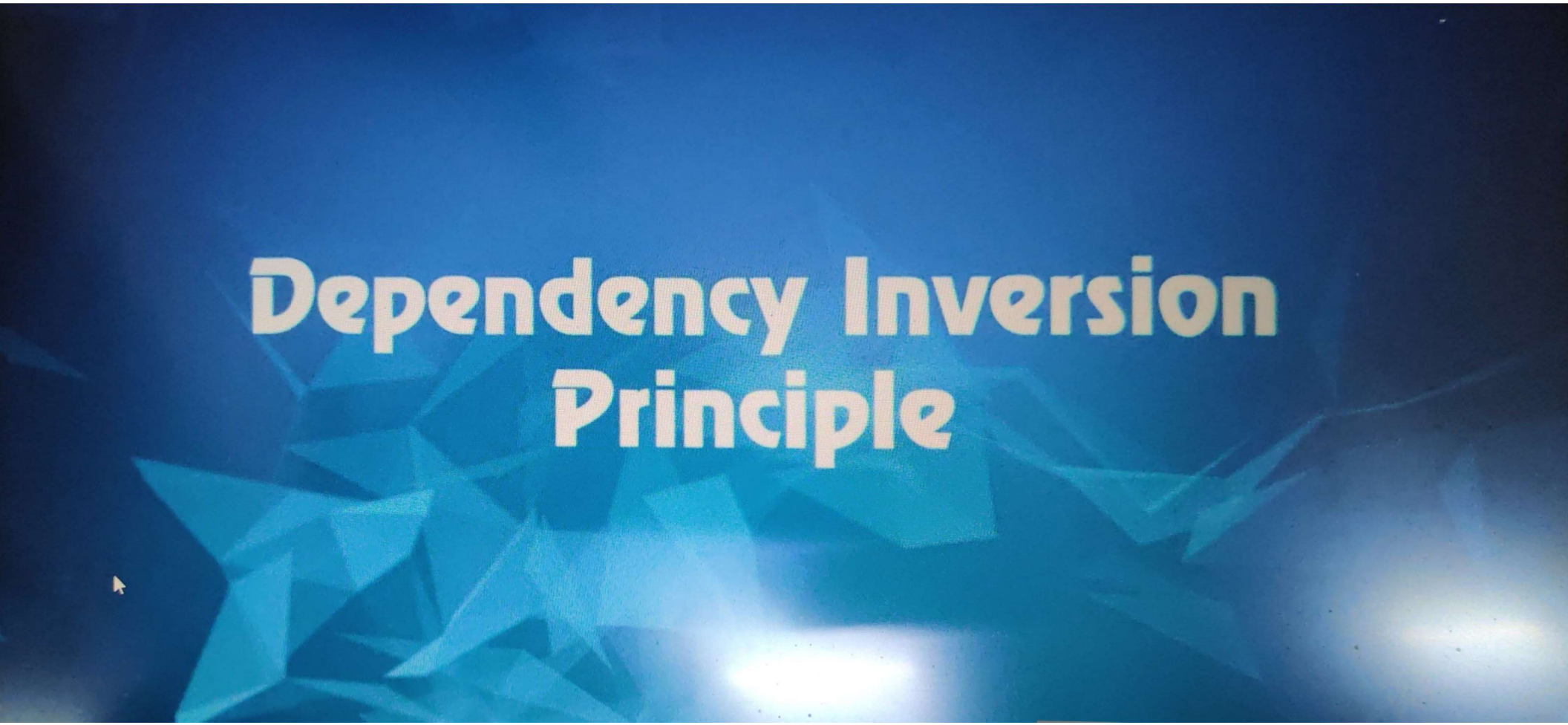

Interface Segregation
Principle


CLIENTS SHOULD NOT BE FORCED TO DEPEND
UPON INTERFACES THAT THEY DO NOT USE

Interface Pollution

Signs of Interface Pollution

- Classes have empty method implementations
- Method implementations throw UnsupportedOperationException (or similar)
- Method implementations return null or default/dummy values


Dependency Inversion
Principle

A. HIGH LEVEL MODULES SHOULD NOT DEPEND UPON LOW LEVEL MODULES. BOTH SHOULD DEPEND UPON ABSTRACTIONS

B. ABSTRACTIONS SHOULD NOT DEPEND UPON DETAILS. DETAILS SHOULD DEPEND UPON ABSTRACTIONS