Solid Principles

Principle: rule or code of conduct.

There are two type of developers

1. The person who thinks short-term, means he thinks solving requirement ASAP with out following any coding stands.
2. If we write code with out following standards in future it may leads problem.

Real world challenges

1. Tightly coupled code.
2. Poor scalability
3. Difficult to understand the code.
4. Bug prone changes.

Code smell:

If we keep on ignoring code smell, then code will become rotting.

Rotting: more bugs, more issues, more difficulty will come in and issues will start, then developer will be in difficult position and developer need to do code re-design.

Code re-design is very difficult process.

To overcome these challenges SOLID principles came into picture

S 🡪 Single responsibility

A class should have only one reason to change.

For single class or any class we should give one responsibility at a time.

O 🡪 Open closed principle

Open for extension but closed for modification.

L 🡪 Liskov substitution principle.

If we have base class, we should be able to any of its subclass in place of the base class without breaking the program.

I 🡪Interface segregation principle.

Client should not be forced or anybody should not to force to use interfaces which there are not meaning to use it.

D 🡪 Dependency inversion principle.

High-level modules should not depend on low-level modules.

SOLID principle going to make application robust enough that it can be extendable, it can be reuseable, bug free, it will be loosely coupled.

These are the benefits going to get by using SOLID principles.

SOLID principles are introduced by uncle bob at 2004.

SOLID principles are nothing but collection of five design guidelines

C0hesion:

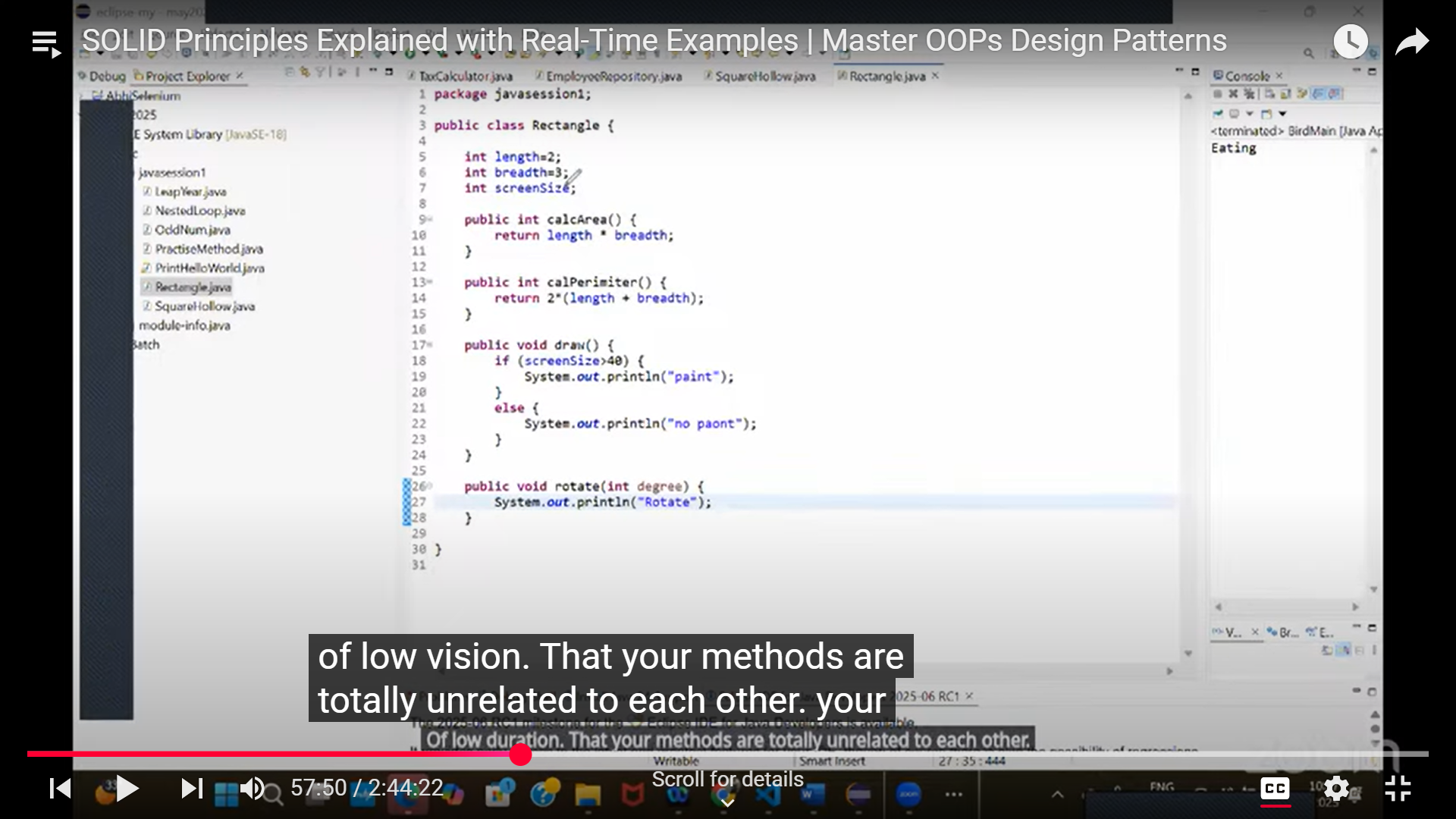
Cohesion means sticking together and stay together. When they stay together, they work together and they work well.

DEF: various parts of software components are related.

Low cohesion:

Even though we unite them they are different different category.

Example:



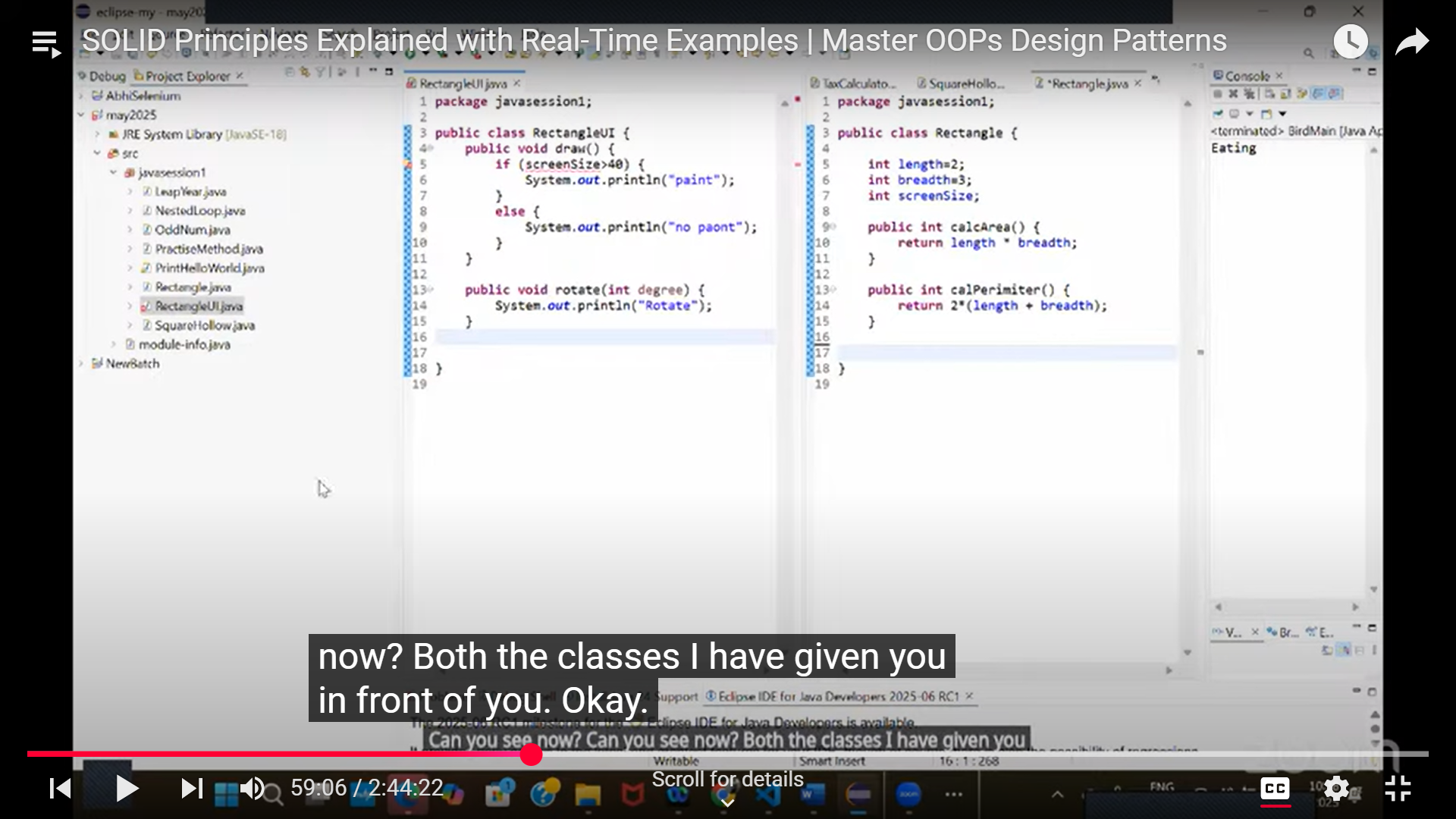
Here calArea() and calPerimeter has related, because it going to write measurements.

But there is no related to above methods and draw() or rotate() methods.

High cohesion:

What ever elements are available, they all are able to mingle together. They all make single sense to you. Unitedly they are working together properly.

Simple term: All the methods are closely related to each other. Don’t keep unrelated methods in same class.



In above example, draw() and rotate() methods are moved to another class.