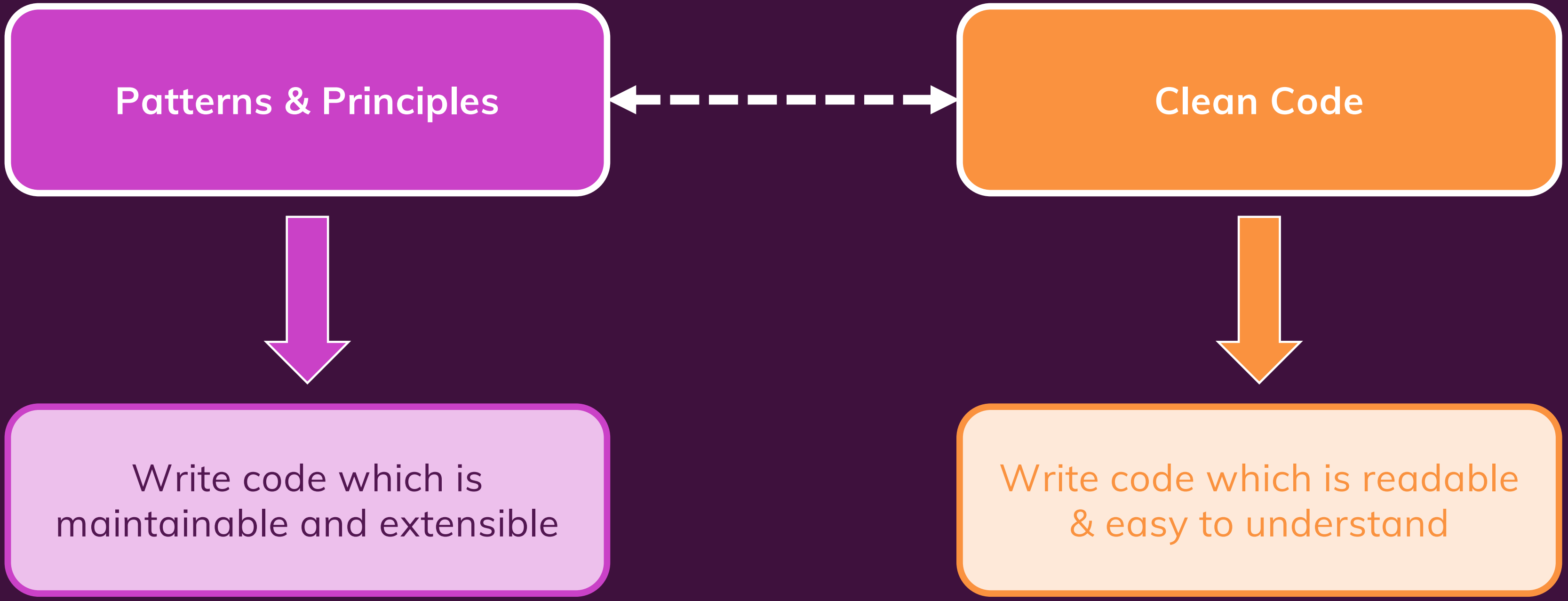
ACADEMIND

This Module Will Not Teach You OOP!

ACADE MIND

We Won’t Dive Into All OOP Patterns & Practices

Clean Code and Patterns & Principles


Patterns & Principles
Clean Code
Write code which is maintainable and extensible
Write code which is readable & easy to understand

The Difference Between Objects & Data Structures

Object	Data Container / Data Structure
Private internals / properties, public API (methods)	Public internals / properties, (almost) no API (methods)
Contain your business logic (in OOP)	Store and transport data
Abstractions over concretions	Concretions only

ACADE MIND

Polymorphism

The ability of an object to take on many forms.

Classes Should Be Small

You typically should prefer many small classes over a few large classes

Classes should have a single responsibility
Single-Responsibility Principle (SRP)

A Product class is responsible for product "issues" (e.g. change the product name)

Cohesion

How much are your class methods using the class properties?

Maximum Cohesion	↔	No Cohesion
All methods each use all properties	Highly cohesive classes	All methods don't use any class properties
A highly cohesive object		Data structure / container with utility methods

Law Of Demeter

```
this.customer.lastPurchase.date;
```

Principle of Least Knowledge: Don’t depend on the internals of “strangers” (other objects which you don’t directly know)

Code in a method may only access direct internals (properties and methods) of:

- the object it belongs to
- objects that are stored in properties of that object
- objects which are received as method parameters
- objects which are created in the method

ACADE MIND

Tell, Don’t Ask!

The SOLID Principles

S	Single Responsibility Principle
O	Open-Closed Principle
L	Liskov Substitution Principle
I	Interface Segregation Principle
D	Dependency Inversion Principle

The Single-Responsibility Principle

Classes should have a single responsibility – a class shouldn’t change for more than one reason.

SRP & Clean Code

Restricting classes to one core responsibility leads to smaller classes

↓

Smaller classes tend to be easier to read

The Open-Closed Principle (OCP)

A class should be open for extension but closed for modification.

OCP & Clean Code

Extensibility ensures small class (instead of growing classes) and can help prevent code duplication (DRY)

$\downarrow$

Smaller classes and DRY code increase readability and maintainability

ACADE MIND

The Liskov Substitution Principle (LP)

Objects should be replaceable with instances of their subclasses without altering the behavior.

The Interface Segregation Principle (ISP)

Many client-specific interfaces are better than one general purpose interface.

The Dependency Inversion Principle (DIP)

You should depend upon abstractions, not concretions.