

W4156

OOAD II: OO Paradigm

Lecture Arc

1: Communicating Designs

How do I communicate the design of a software design/architecture?

2: Object Oriented Analysis and Design

What is OOAD and how does it help translate a problem domain into a technology solution?

3: Patterns

Are there recurring problems and a 'library' of customizable 'good' solutions?

Can I avoid solving every solution from scratch?

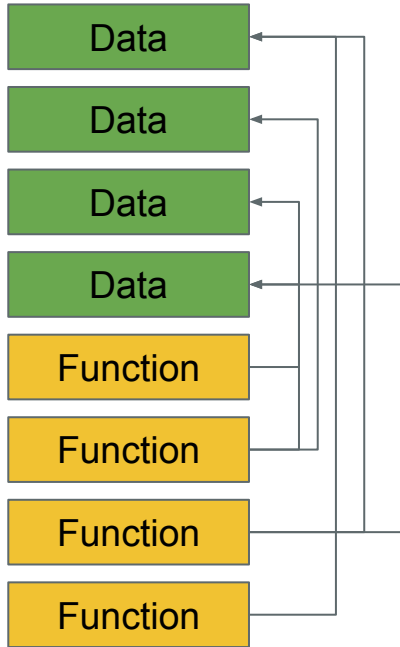
Agenda

Dig deeper into understanding OOAD

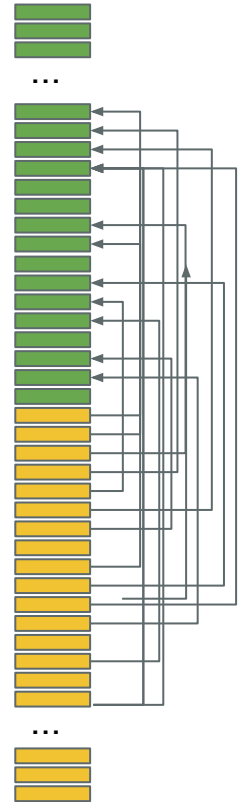
- ❑ What problem creates need for OOAD?
- ❑ What is OOAD?
- ❑ Key Principles
 - ❑ Classes
 - ❑ Relationships

Problem: Code Soup

What is going on?



- Is this coding paradigm easy to:
 - Read,
 - Test
 - Change/Extend
 - Reuse
- Does it scale with complexity?



Complexity (yet again)

Yet again we are in a fight to master complexity

As computer scientists we have a small set of trusty tools

- **Abstraction:** we build abstraction on abstraction to hide complexity
- **Encapsulation:** we hide complexity within abstractions to build higher order primitives

Ok - what is OOAD?

Enter OOAD

A **paradigm** (way of thinking and designing)

1. **Analyze** our *problem domains* in terms of object interacting with objects
2. **Design** our program as series of interacting objects
3. Use programming languages/features supporting '**objects**'

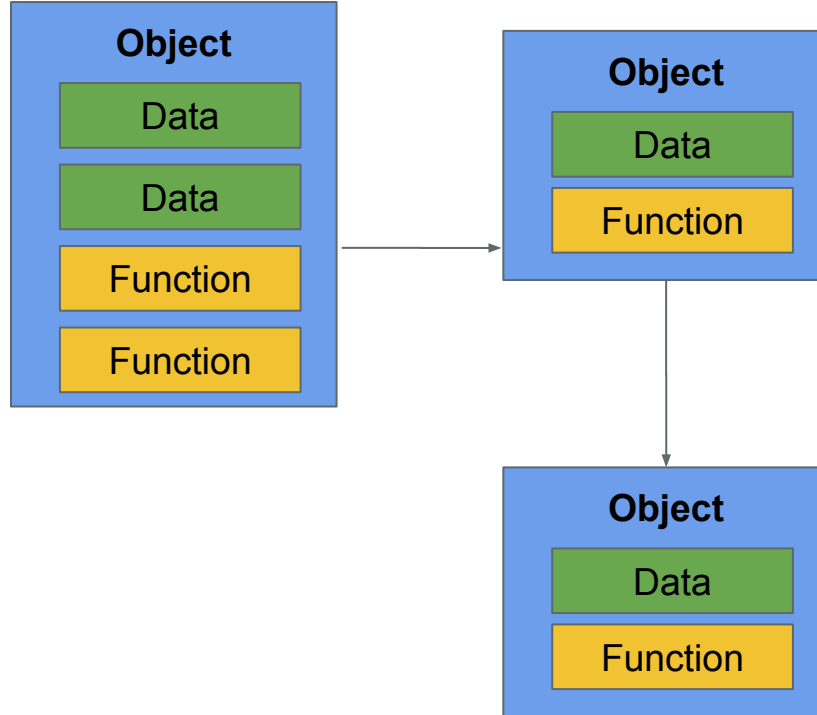


Paradigm yields¹ **benefits** of:

- Encapsulation
- Abstraction / Controlling Complexity
- Testability, Readability, Modifiability

¹ I will present opposing points of view later

Object Oriented Version (over simplified)



Why is it called Object Oriented?

1. **Everything** is an object
2. Objects communicate by sending and receiving messages
[messages are also objects. See #1]
3. Objects have their own memory
[memory is made of objects. See #1]
4. Every object is an instance of a class
[classes are also objects! See #1]
5. The class holds the shared behavior for its instances
[classes are also objects! See #1]
6. To run a program pass control to the first object
[Everything is then messages between objects. See #1]



In all seriousness - for repetition. OO is:

1. a *mindset* to view/'see' the world as objects interacting with objects
2. to model software solutions in this paradigm
3. to code in language constructs in the paradigm

(I didn't even need to make these ones!)

Example 1: What is the OO model of Ewans journey to deliver this lecture and deliver the lecture itself¹?

1. What are the objects?
2. For each object:
 - What data do they hold?
 - What can each object *do/functions*?
3. What are the interaction between objects (what messages/objects are exchanged)
4. Can we describe the relationship between objects in terms of
 - a. “X is a Y”
 - b. “X has a Y”
 - c. “X uses a Y”
5. Do we need to model atoms/oxygen? When do we stop?

¹(I am prepared to be the butt of as many jokes as required for you to grasp OO)

Example 2:

Can we refactor the previous example in OO paradigm?

(We will do this crudely/intuitively today. In the next lecture we will discuss the *analysis and design process/techniques*)

Key Principles

Deriving OO Constructs from Needs

How do I ...	Concept
manage complexity / what is my primary paradigm construct?	Object
store the 'data'/'state' of objects?	<u>Attributes</u> (encapsulation)
represent the 'behavior' of objects?	<u>Methods</u> (encapsulation)
provide encapsulation / limit exposure of methods/attributes	<u>Visibility</u> (second <u>link</u>)
recognize/represent objects may be same 'entity' share similar data/behavior	<u>Class (and instance)</u>
represent more complex relationships between classes	<u>Relationship between classes</u>
represent classes that should not be able to be 'instantiated'	<u>Abstract</u>
represent methods or data that should be <i>shared</i> across all instances	<u>Static</u> Method and <u>Scope</u> Variable
establish initial state/behavior of objects?	<u>Constructors</u> and <u>here</u>
work out if two of my objects 'the same'	<u>Equality</u> , <u>Hash</u>
test if an object supports a specific behavior	<u>Instanceof</u>

Relationships Between Classes

Type	Description	Example
Association	Uses	Car uses road
Inheritance	Is a	Professor is a human being
Aggregation	Weak containment	Grouping. Child can exist independent. Delete class and students still exist!
Composition	Strong containment	Real world whole part. Child not independent. Delete house deletes rooms!

Breathe

Remember:

1. Think ‘real world \Rightarrow code’ rather than ‘code \Rightarrow real world’
2. All of the language features are to help ‘represent/model’ our ‘problem domain/the real world’ in code
3. If we lacked these constructs we would find it hard to represent the inherent structure of our problem domain in code

Real World » Code

i.e. below are ‘axiomatic facts’ of the real world (they are true things we need to somehow capture and model)

- Within a university: Faculty and Students are both people but:
 - Who can grade?”
 - Who has a payroll ID?
 - Who has a GPA?
- Within a store
 - Products have prices (‘can of soup is \$2.99)
 - If an *individual* can is bashed it gets its own discounted price
 - Products can go on sale
- Convert mph to kph is a method but not part of an instance of car?

Second OO Example / Building it Up ...

Warning:

- OO is often taught using 'physical' examples (Cat, Dog, Chicken, Animal, Mammal)
- This is great to relate to *initially*
- However, they miss other abstractions/concepts

Example:

- AirBnB: people, admins, renter, lister -> (all physical)
- Perhaps we identify a 'booking' -> (tangible but not physical)
- How do we validate an entry? -> (abstract activity - 'Validator')
- Are we able to cancel a booking? -> (more abstract 'CancellationPolicy')
- *Process/workflow* to verify a host? -> (more abstract 'VerificationWorkflow')

Okay

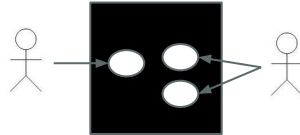
Next Lectures: Process of Problem ➤ Code

We are trying to
design a system
(currently
“fuzzy/unclear”)

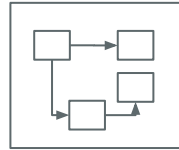
What does it
need to do?
(scope and
functions)

Conceptual
model

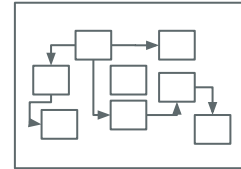
Implementable
Design



(use-cases)



(conceptual class diagram)



(implementation level
class diagram)

Pop Quiz

Question	Answer
What is the motivation for object oriented design?	
What are the key tenets of the paradigm?	
In OO everything I try to model my problem domain as <X> and the <Y> between <X>	
What is the 'AD' in 'OOAD?'	
Ewan is an <> which is an instance of <>	
When we say Python supports object orientation what do we mean?	

Reading

Reading	Optionality
<u>Springer OOAD</u> (chapters 1,2,3 & + 4)	Required (spread over design lectures) [I gave 1-3 before midterm but am relisting as a reminder!]
<u>Python Support</u> for OO	Required