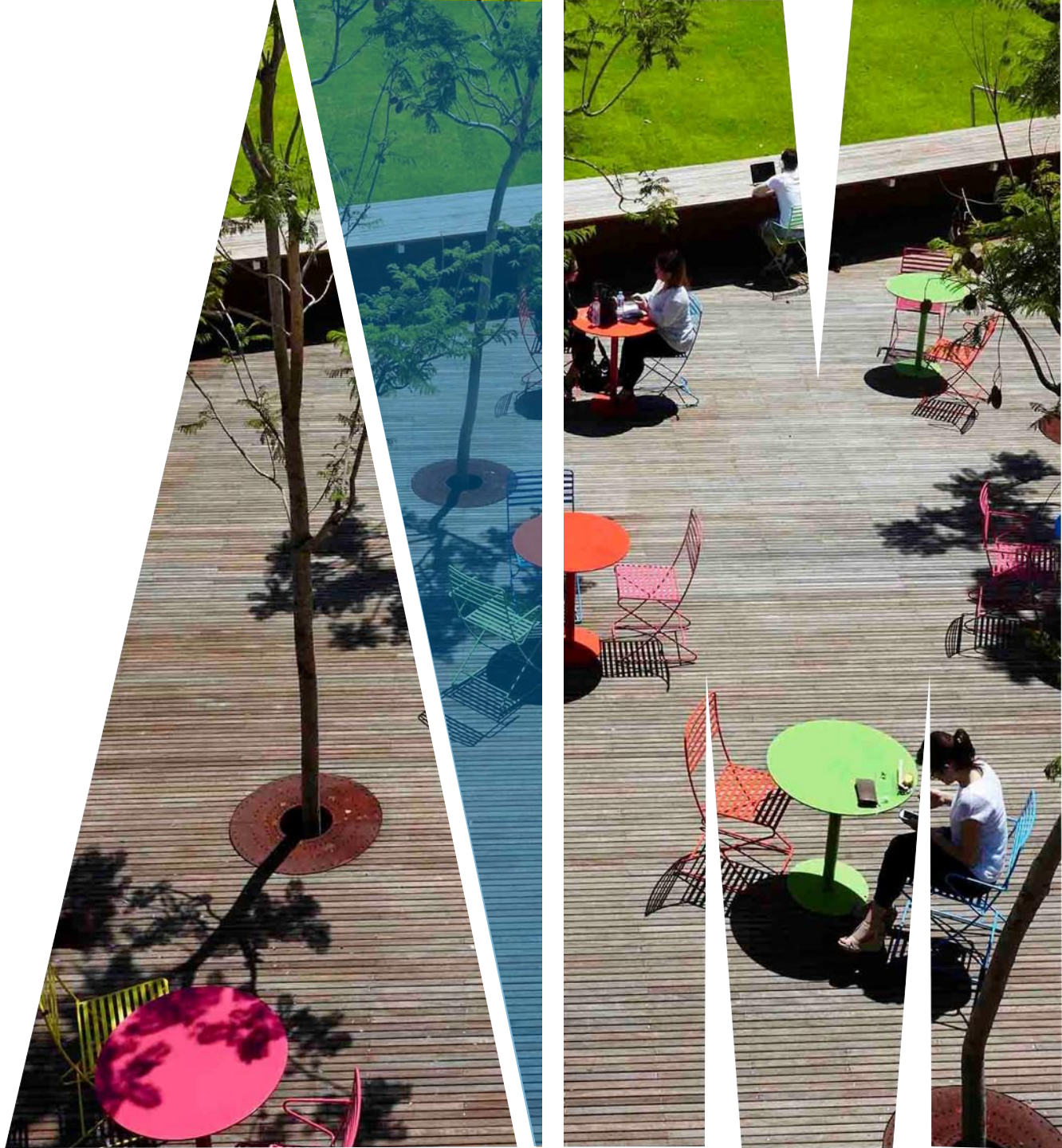
MONASH University

FIT2099 Object-Oriented Design and Implementation

Review of
OO Design Concepts



MONASH University

WHAT IS ABSTRACTION?

According to dictionary.com,

> *"the act of considering something as **a general quality or characteristic**, apart from concrete realities, specific objects, or actual instances."*

To a software developer, this means deciding

- **what information** do we need in order to represent some item or concept?
- **what should we expose to the rest of the code** (i.e. make public) so that we will be able to use this part easily?

WHAT IS SEPARATION OF CONCERNS

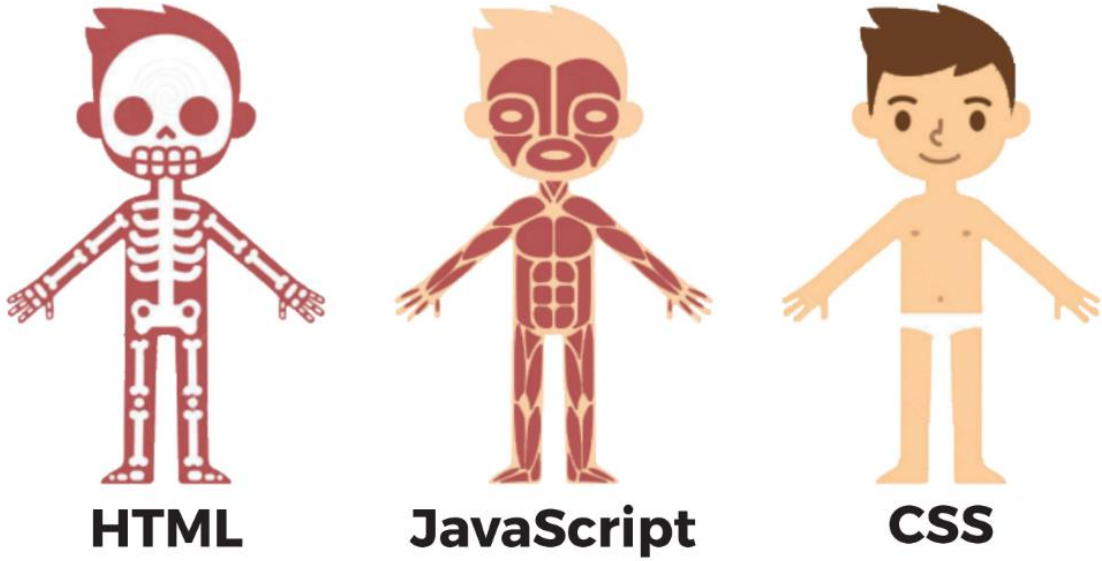
For our purposes, a **“concern” is a responsibility**

Every ‘module’ should have a **single, well-defined** set of responsibilities

Responsibilities should **overlap as little as possible** with other ‘modules’

- shared responsibilities often leads to repeated code

Unclear responsibilities make the ‘module’ **hard to use**.



WHAT IS ENCAPSULATION?

The idea that a module has an outside that is distinct from its inside, that it has an **external interface** and an **internal implementation**

cf. data abstraction, information hiding

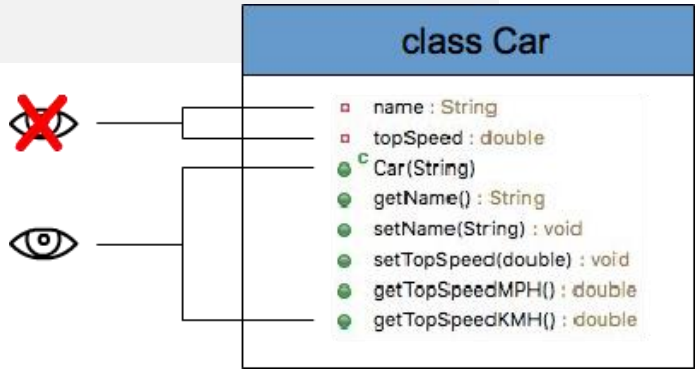
--IEEE Software Engineering Vocabulary

WHAT IS INFORMATION HIDING?

Every module is characterized by its **knowledge of a design decision which it hides from all others**. Its *interface* or *definition* was chosen to reveal as little as possible about its inner workings: data structures, its internal linkings, accessing procedures and modifying procedures are part of a single module.
-- David Parnas (**1972**)

Information/implementation hiding is the use of encapsulation to **restrict from external visibility** certain information or implementation decisions that are internal to the encapsulation structure.

-- Meilir Page-Jones



OTHER RELATED CONCEPTS: ENCAPSULATION AND INFORMATION HIDING

Here's a guide:

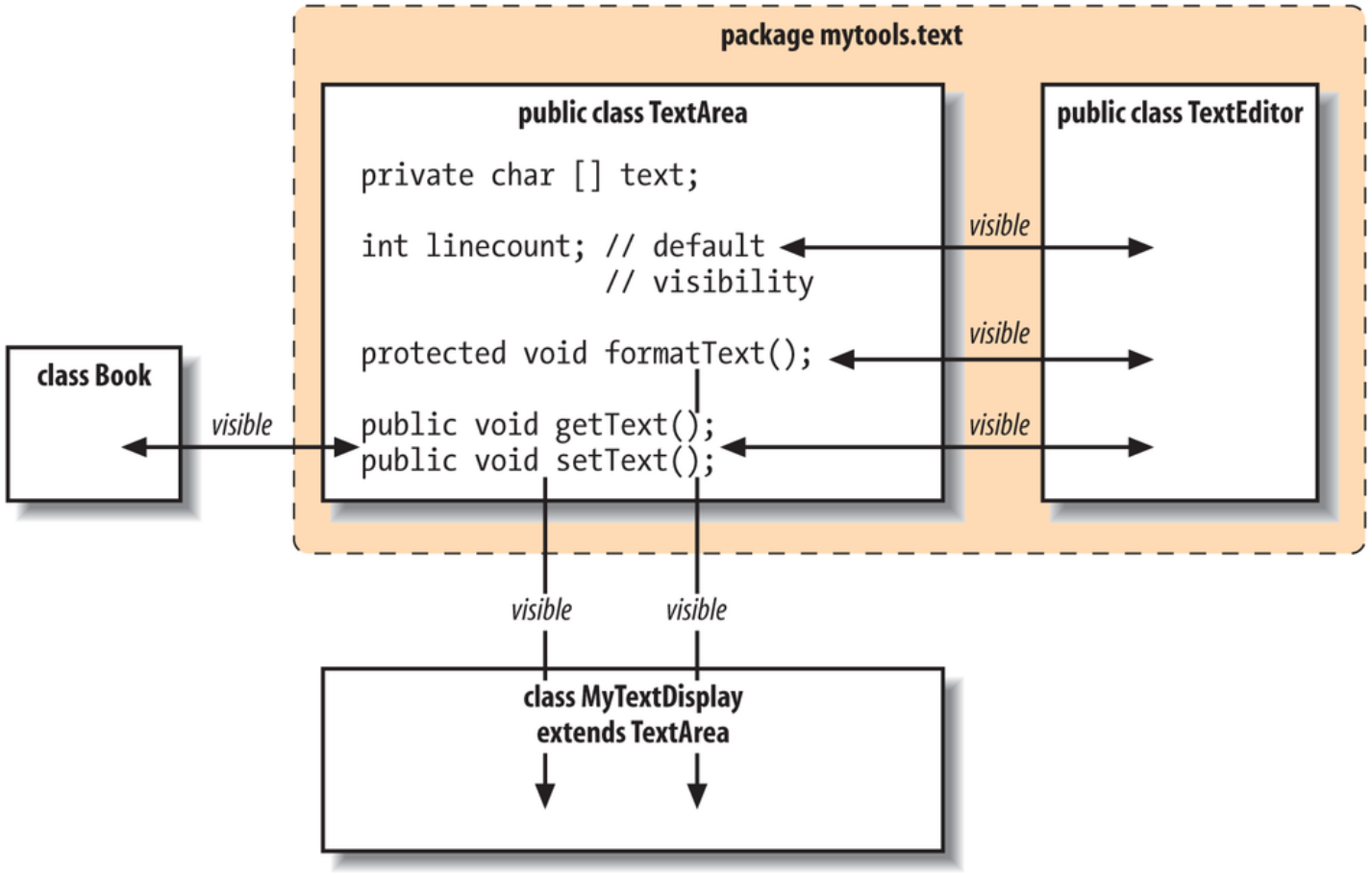
- we use **encapsulation** when we bundle things together
- we use **abstraction** when we decide which things should be bundled together
  - we also use **abstraction** when we decide how things should look from outside (i.e. when we design a class's public interface)
- we use **information hiding** whenever we use an encapsulation mechanism that doesn't allow access from outside
  - private or protected modifiers: keep implementation details hidden
  - local variables: no access from outside the method
  - defensive copying: prevent external code from accessing internal data structures

ENCAPSULATION
BOUNDARIES

Any method call or attribute accesses that that is not in the same class (or package) **crosses** an encapsulation boundary

You want to **minimize** these accesses - that's what we mean by "ReD"

So… expose (i.e. make public) the methods/attributes that client code really needs, and hide everything else



WHAT IS POLYMORPHISM?

Polymorphism in OOP can be broadly described as **the ability of a message to be displayed in more than one form**.

Another way to define it:
The ability of **performing a single action in different ways.**

In practical (aka. coding) terms:
The ability of **defining one interface** and have **multiple implementations.**

ABSTRACT CLASSES

REAL-WORLD REPRESENTATION

Animal

Speak() *How does a (specific animal) sound?*

Oink oink!

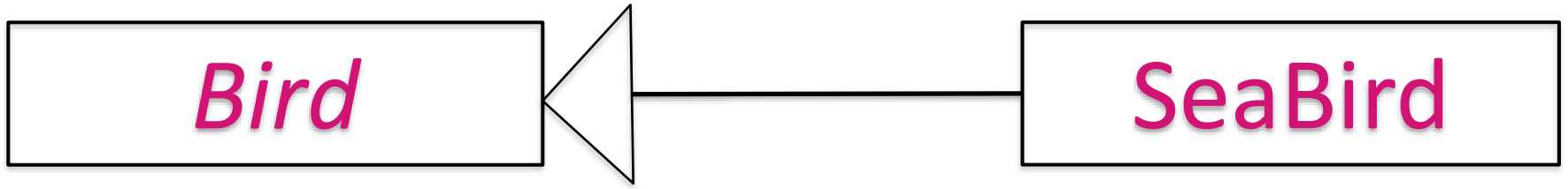
Miau miau!

Woof woof!

Quack quack!

INTERFACES

REAL-WORLD REPRESENTATION



Seagull

Can swim and fly

Frigatebird

Can fly but cannot swim, they drawn if they fall into the water

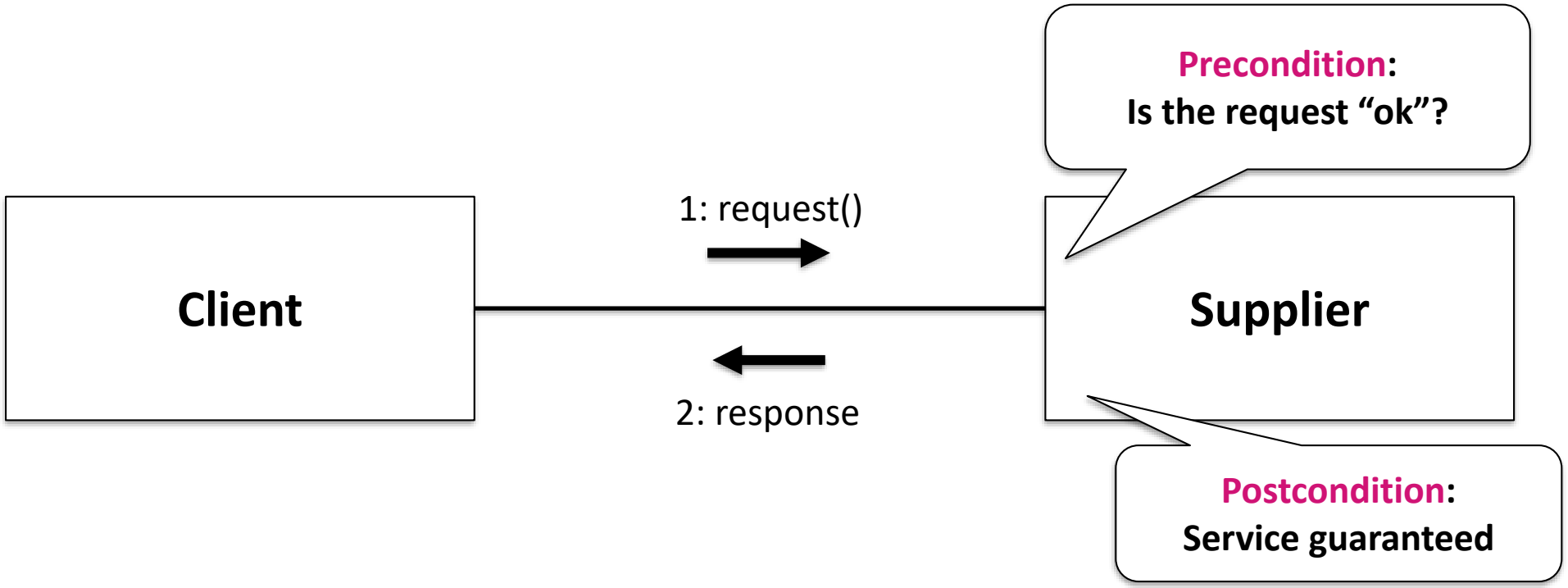
Penguin

Can swim but, definitively, cannot fly.

CONCRETE/ABSTRACT V/S INTERFACES

	Interfaces	Abstract classes	Concrete classes
Constructor	✗	✓	✓
Static/final attributes	✓	✓	✓
Non-static or non-final attributes	✗	✓	✓
PRIVATE attributes and methods	✗	✓	✓
PROTECTED attributes and methods	✗	✓	✓
PUBLIC methods	✓	✓	✓
ABSTRACT methods	✓	✓	✗
STATIC methods	✓	✓	✓
FINAL methods	✗	✓	✓
DEFAULT methods	✓	✗	✗
Multiple inheritance?	✓	✗	✗

THE CLIENT AND SUPPLIER (Design by Contract)



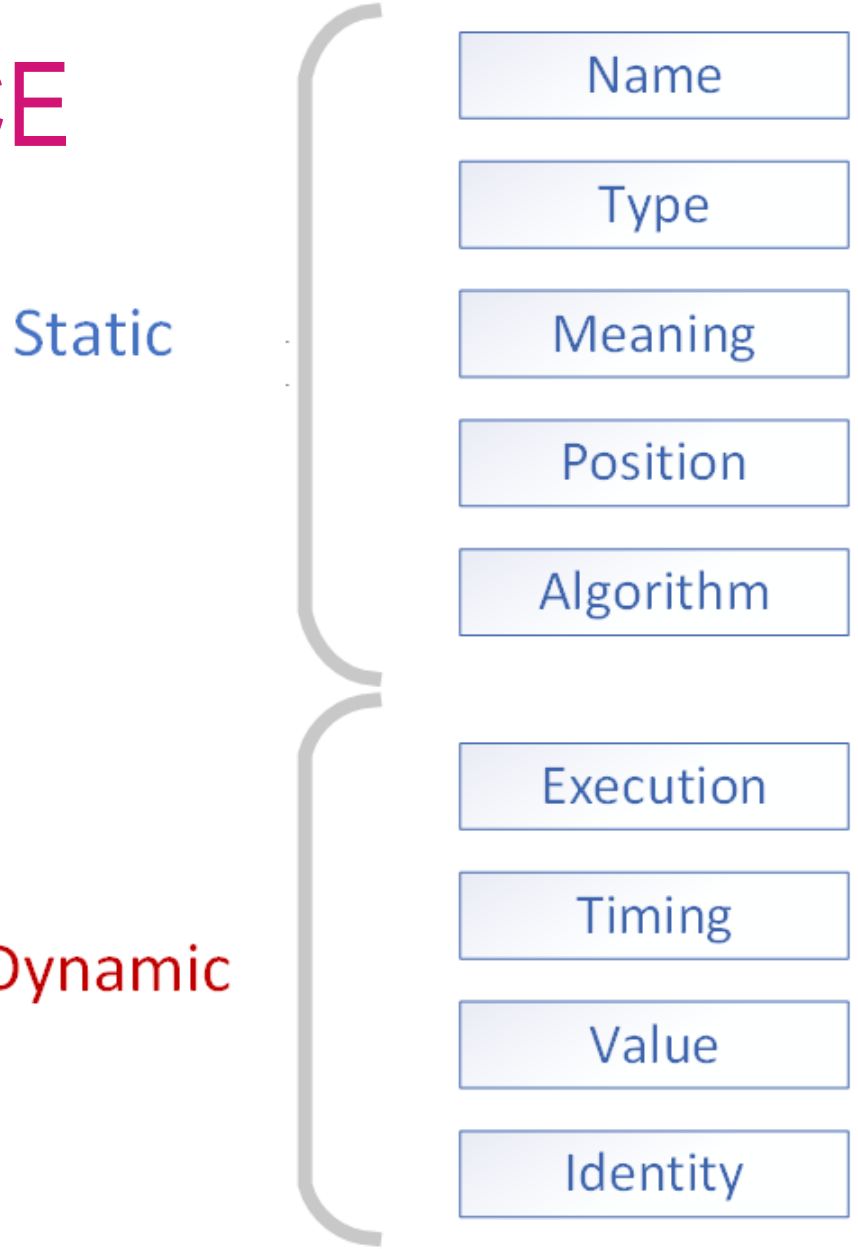
WHAT IS CONNASCENCE?

*I say that two elements of software are connascent if they are "**born together**" in the sense that they somehow share the same destiny.*

*More explicitly, I define two software elements A and B to be connascent if there is **at least one change that could be made to A that would necessitate a change to B** in order to preserve overall correctness.*

– Meilir Page-Jones

LEVELS OF CONNASCENCE



WHEN DOES A PRIVACY LEAK OCCUR?

When getters return a reference to a private object that is *mutable*

i.e. with public attributes or mutator methods other than constructor

Generally, you should **make a copy and return that.**

Otherwise, you lose benefit of encapsulation

this is called a ***privacy leak***

Lose control of connascence

THE CODE SMELLS

A code smell is a surface indication that usually corresponds to a deeper problem in the system

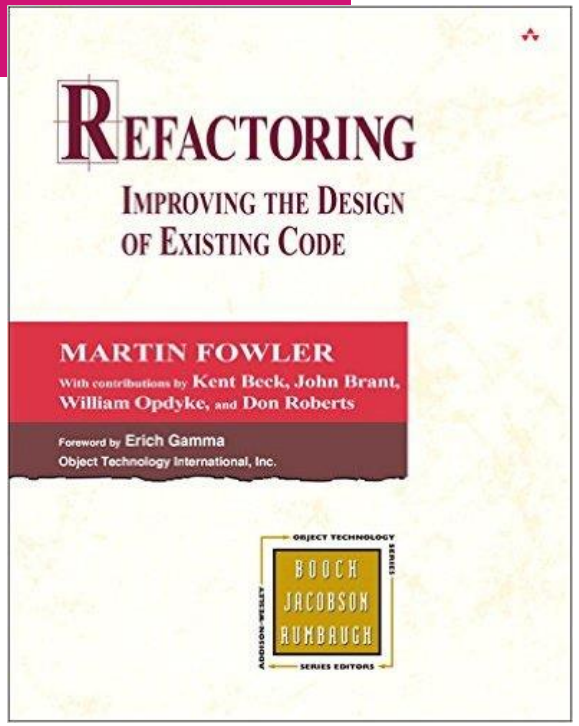
– Martin Fowler

A deeper problem is usually a **design problem**

By extension, “design smells” are some small bits of a design that commonly indicate a broader problem.

WHAT WAS REFACTORING, AGAIN?

…is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.
– Martin Fowler



Refactor to improve the quality of software
Fowler presents a *technique* for refactoring
Book is available from the library
- get second edition if you can find it

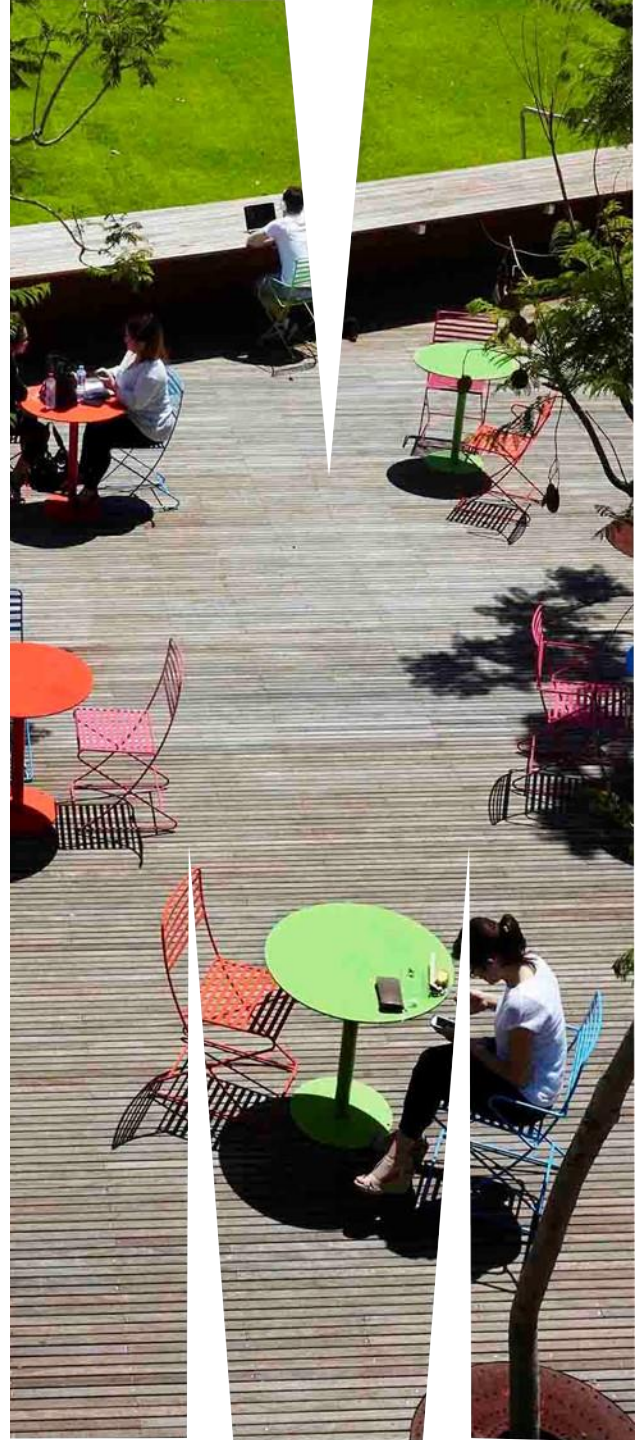
Summary

Reflect on the work you have done in this unit

How were these concepts reflected in
the UML diagrams and code you created?

MONASH University

Thanks



MONASH University