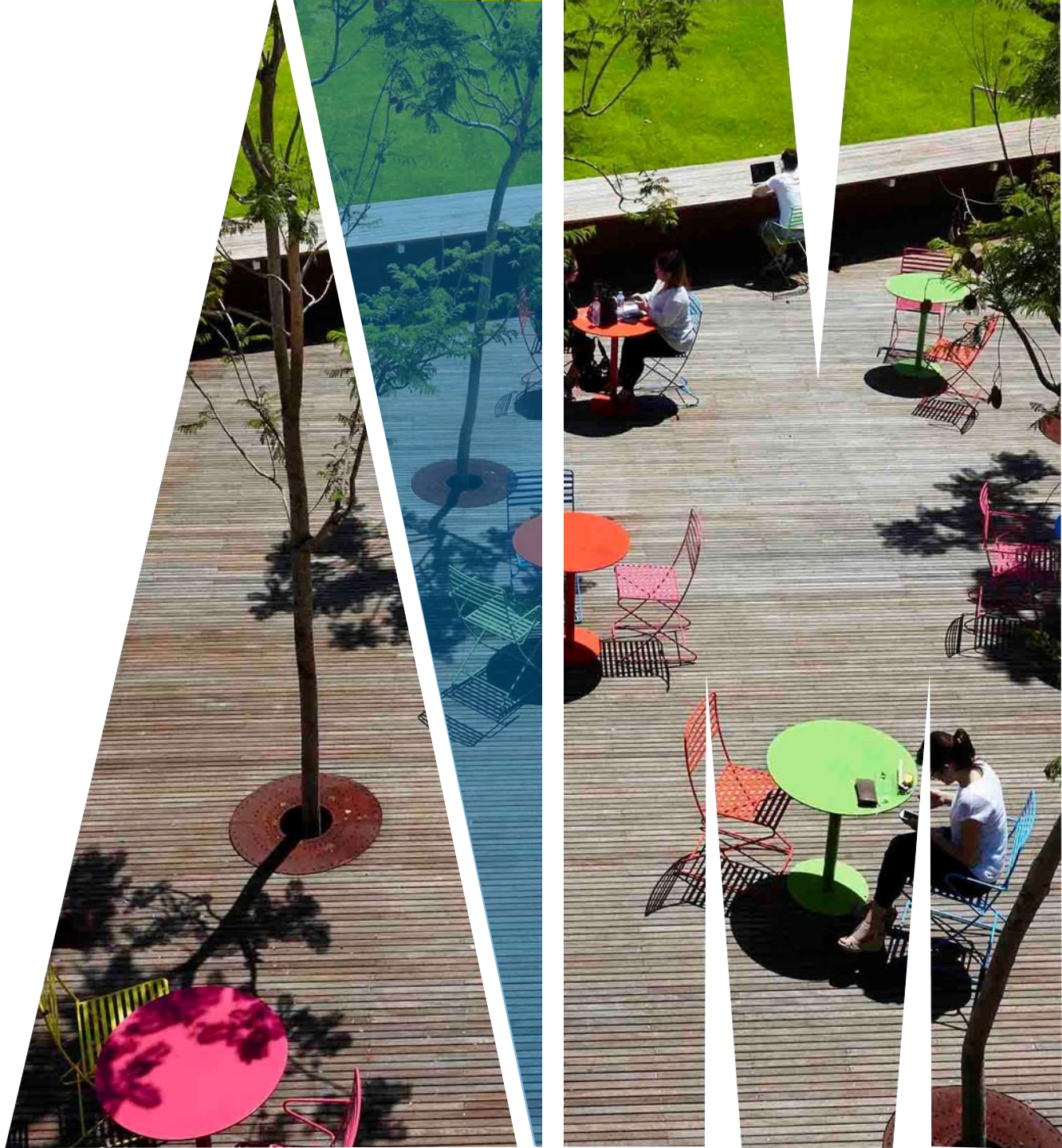
MONASH University

FIT2099 Object-Oriented Design and Implementation

Introduction to Connascence: Dependency Control



MONASH University

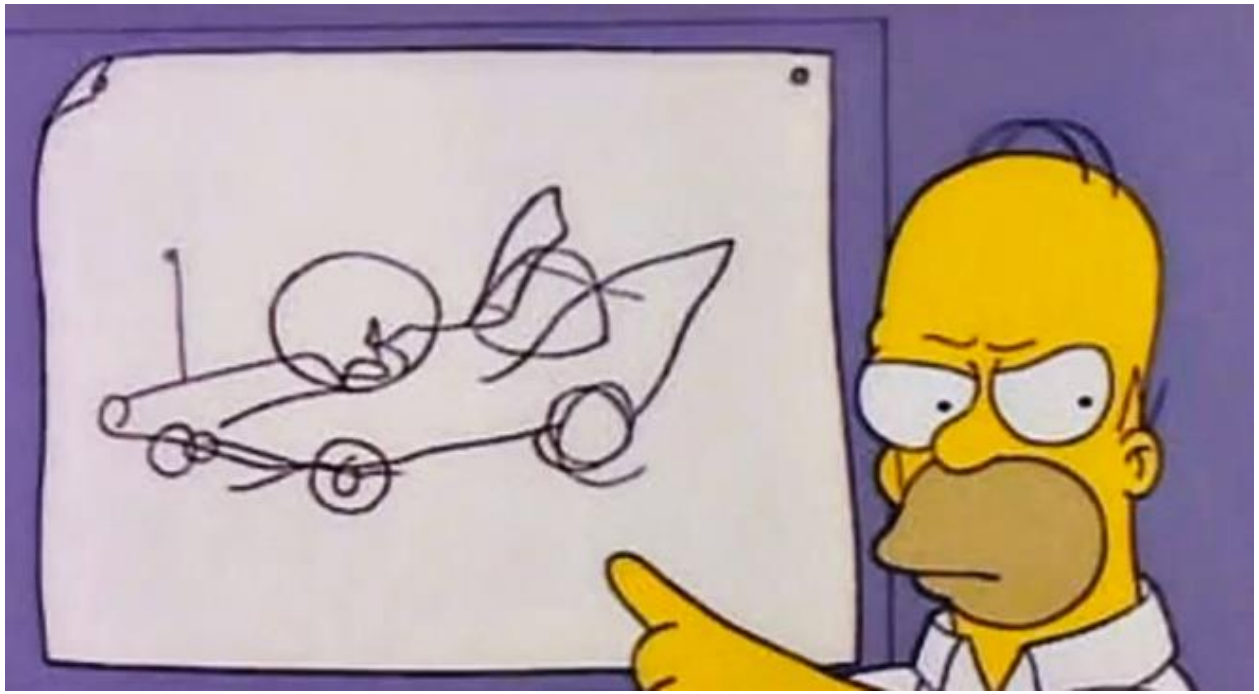
Outline

Good and bad designs

Quality of design

Dependency control

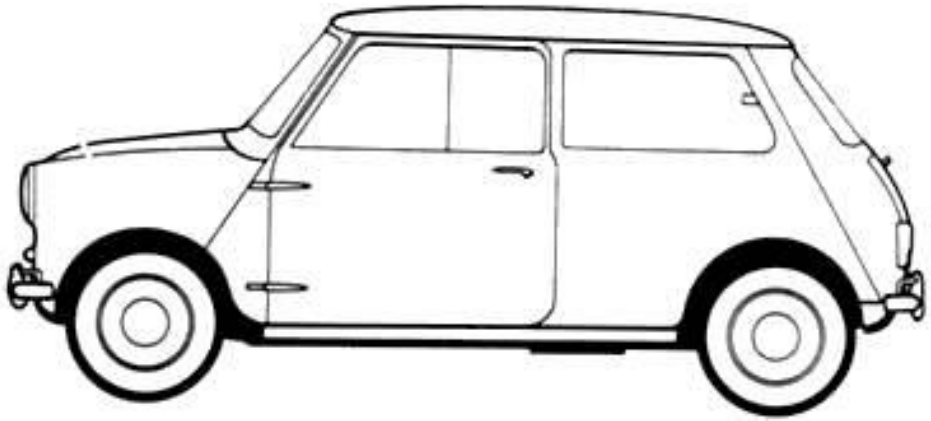
A BAD DESIGN LEADS TO A BAD PRODUCT



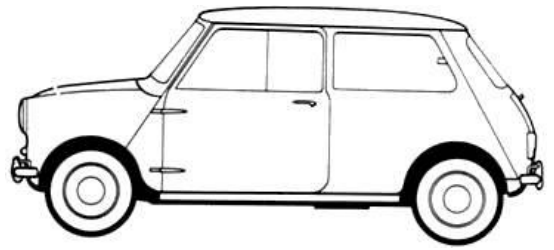
A BAD DESIGN LEADS TO A BAD PRODUCT

MONASH University

A GOOD DESIGN CAN LEAD TO A SUCCESFUL PRODUCT



WHAT ARE OUR AIMS IN SOFTWARE DEVELOPMENT?



Working software

On **time**

On **budget**

BUT…

“Working software” isn’t just about adding features

must also be **usable**

must be **able to be fixed/extended** as requirements change (even after deployment)

“On time” and “on budget” will only happen if software isn’t too hard to construct

The Mini Minor wasn’t feature-packed, but was very **cheap**, **robust**, and **easy to build** and **maintain**

GOOD DESIGN ≠ “LOTS OF FEATURES”

When most users want this...

...it's not a good idea to ship this

GOOD DESIGN ≠ "LOTS OF FEATURES"

Consumers *do* want features – BUT:

- must be **easy to use/access**
- must work **reliably**
- must be **easy to fix** when necessary
- must be **relevant** to consumer's needs
- must **not cost too much**
- must **not take too much time to implement**

WHAT IS GOOD DESIGN IN SOFTWARE?

Some combination of:

- functionally correct
- performs well enough
- usable
- reliable
- maintainable

Exactly which of these is most important depends on the system and its **context**

These are properties of the ***system itself***, not any design artefacts

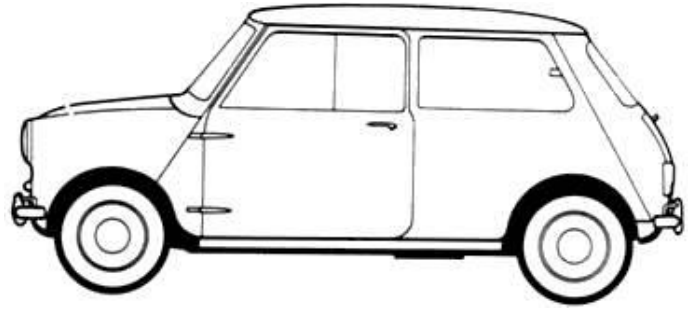
FORMAL CHARACTERISTICS OF GOOD SOFTWARE DESIGN

Characteristic	Definition
Simplicity	The amount and type of software elements needed to solve a problem
Coupling	Measure of interdependence between two or more modules
Cohesion	The degree to which a software module is strongly related and focused in its responsibilities
Information hiding	A component encapsulates its behaviors and data, hiding the implementation details from other components
Performance	The analysis of an algorithm to determine its performance in terms of time (i.e., speed) and space (i.e., memory usage).
Security	A set of technical controls intended to protect and defend information and information systems

Voorhees D.P. (2020) **Characteristics of Good Software Design**. In: Guide to Efficient Software Design. Texts in Computer Science. Springer, Cham. https://doi.org/10.1007/978-3-030-28501-2_11 (open resource)

FORMAL CHARACTERISTICS OF GOOD SOFTWARE DESIGN: SIMPLICITY

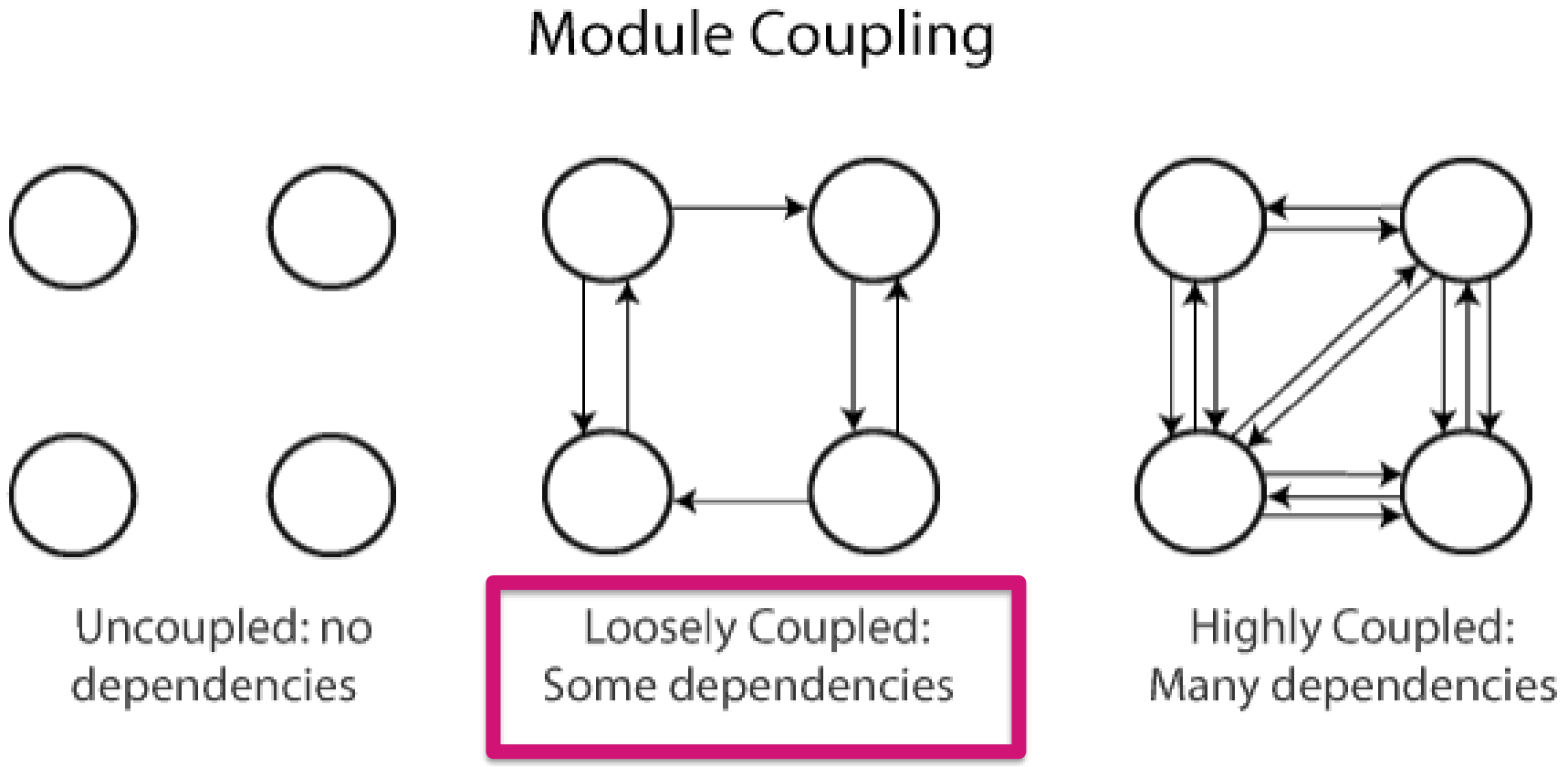
Characteristic	Definition
Simplicity	The amount and type of software **elements needed to solve a problem**



Voorhees D.P. (2020) **Characteristics of Good Software Design**. In: Guide to Efficient Software Design. Texts in Computer Science. Springer, Cham. https://doi.org/10.1007/978-3-030-28501-2_11 (open resource)

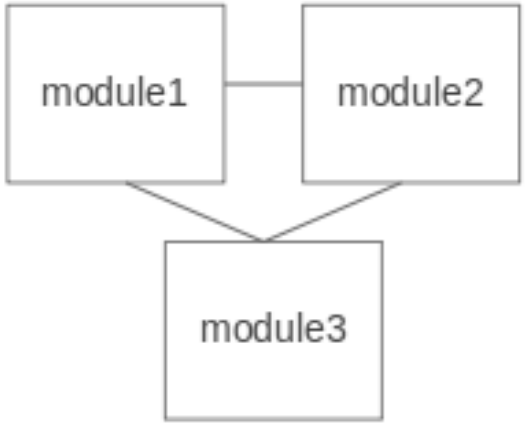
FORMAL CHARACTERISTICS OF GOOD SOFTWARE DESIGN: COUPLING

Characteristic	Definition
Coupling	Measure of **interdependence** between two or more modules **(for example, classes)**

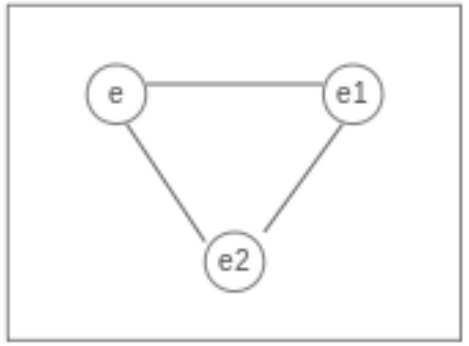


FORMAL CHARACTERISTICS OF GOOD SOFTWARE DESIGN: COHESION

Characteristic	Definition
Cohesion	The degree to which a software module **(e.g. a class)** is strongly related and **focused** in its responsibilities



Vs



Module

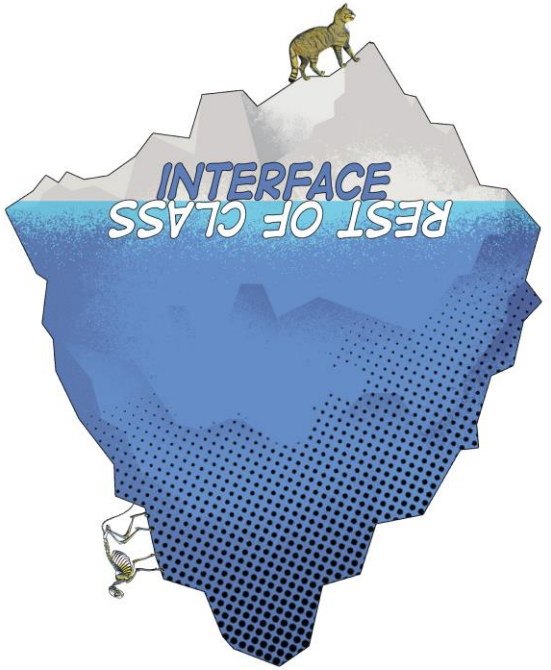
Coupling

Cohesion

CHARACTERISTICS OF GOOD SOFTWARE DESIGN

INFORMATION HIDING

Characteristic	Definition
Information hiding	A component **encapsulates** its behaviors and data, **hiding the implementation** details from other components



FORMAL CHARACTERISTICS OF GOOD SOFTWARE DESIGN: PERFORMANCE

Characteristic	Definition
Performance	The analysis of an algorithm to determine its performance in terms of time (i.e., **speed**) and space (i.e., **memory usage**).

FORMAL CHARACTERISTICS OF GOOD SOFTWARE DESIGN: PERFORMANCE

Characteristic	Definition
Security	A set of technical controls intended to **protect and defend information** and information systems

FORMAL CHARACTERISTICS OF GOOD SOFTWARE DESIGN

Characteristic	Definition
Simplicity	The amount and type of software elements needed to solve a problem
Coupling	Measure of interdependence between two or more modules
Cohesion	The degree to which a software module is strongly related and focused in its responsibilities
Information hiding	A component encapsulates its behaviors and data, hiding the implementation details from other components
Performance	The analysis of an algorithm to determine its performance in terms of time (i.e., speed) and space (i.e., memory usage).
Security	A set of technical controls intended to protect and defend information and information systems

Voorhees D.P. (2020) **Characteristics of Good Software Design**. In: Guide to Efficient Software Design. Texts in Computer Science. Springer, Cham. https://doi.org/10.1007/978-3-030-28501-2_11 (open resource)

HOW CAN CREATE A "GOOD DESIGN"?

Unfortunately, there is no algorithm that will always create a good design

some people will tell you that there is

these people are wrong

There is no algorithm that is guaranteed to *identify* a good design

Over the years, key **principles** have been identified

Today, we will look at one of the most important: **the need to control dependencies**

DEPENDENCY CONTROL

ReD

Biggest issue in design

Controlling the **extent** of dependencies

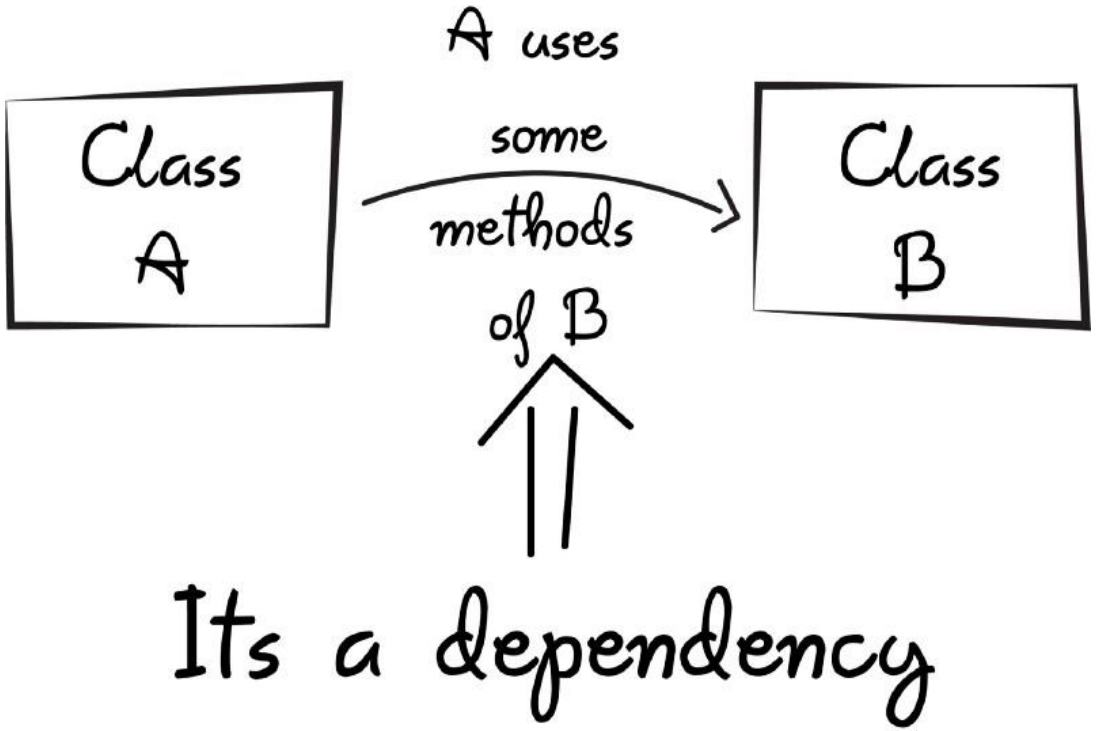
Controlling the **nature** of dependencies

WHAT ARE DEPENDENCIES, AGAIN?

Dependencies are **unavoidable**

If code unit A depends on code unit B:

- **bugs** in B may manifest in A
- **changes** to B may require changes to A

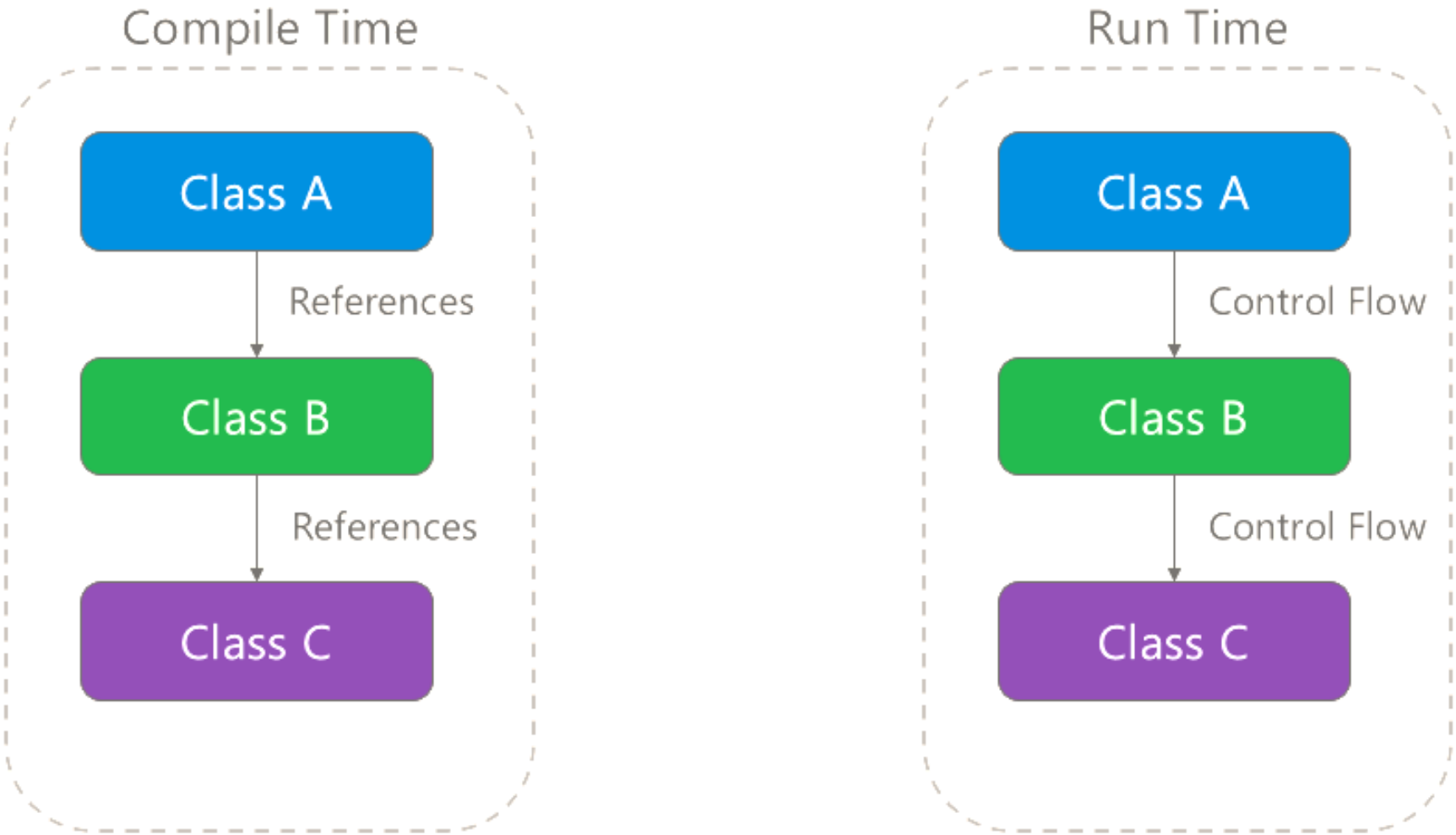


WHAT TO DO WITH DEPENDENCIES?

So we want dependencies to be:

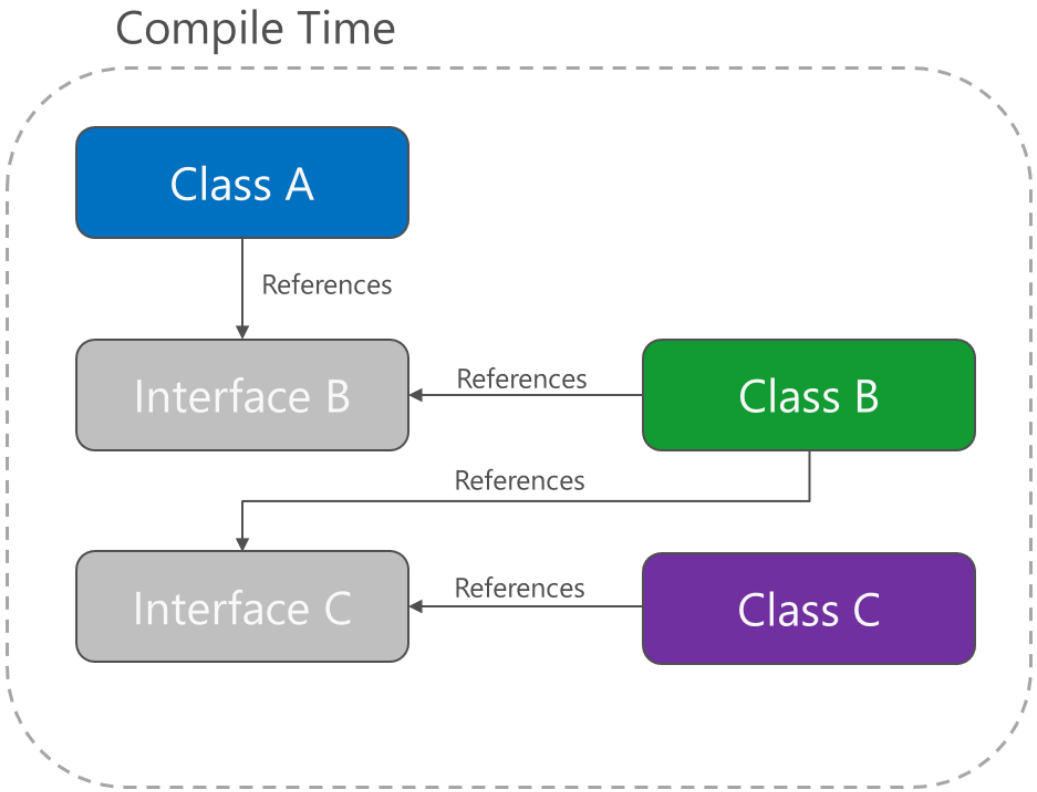
- **only present where necessary**
- **explicit**
- **easy to understand**

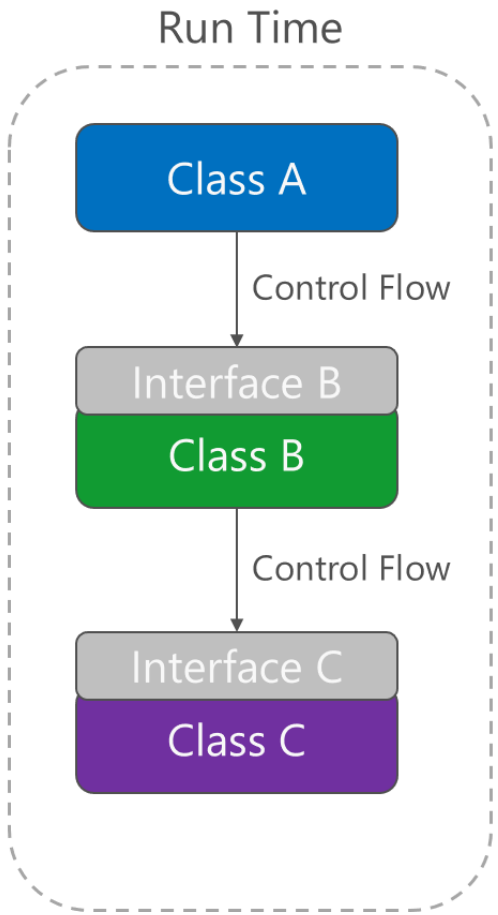
DIRECT
DEPENDENCIES



INVERTED DEPENDENCIES

Using **abstraction**





Summary

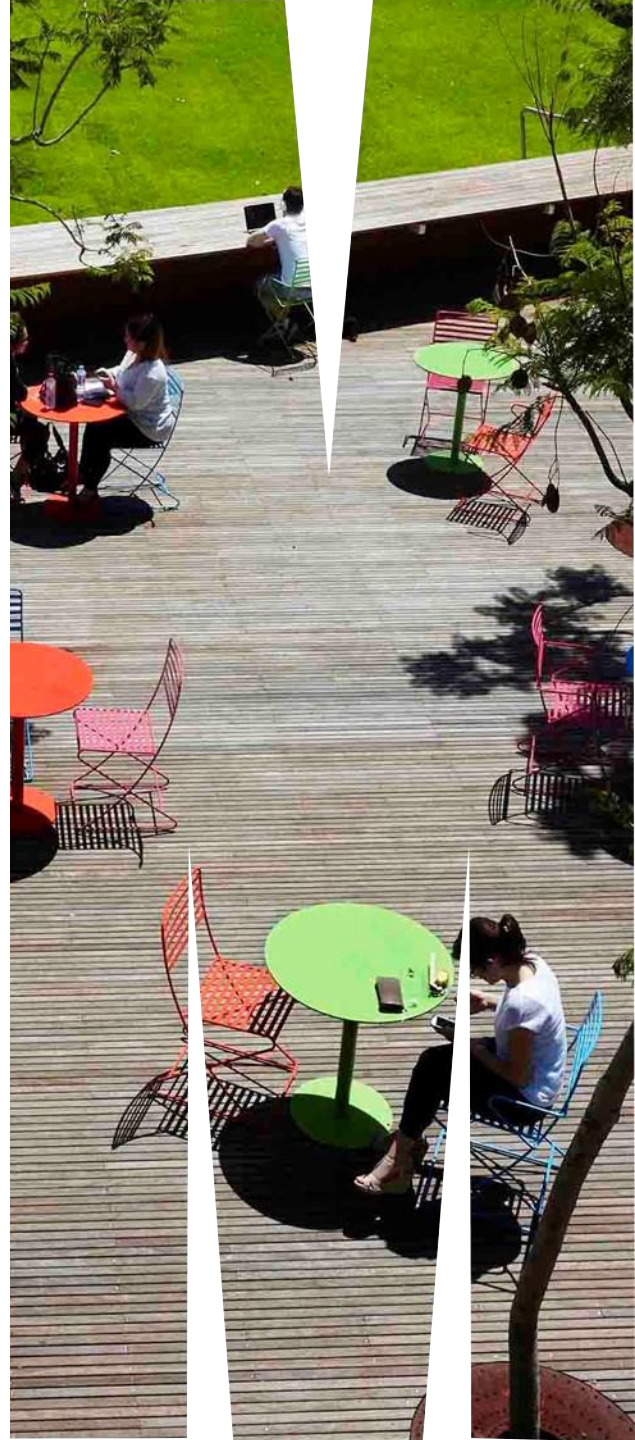
Good and bad designs

Quality of design

Dependency control

MONASH University

Thanks



MONASH University