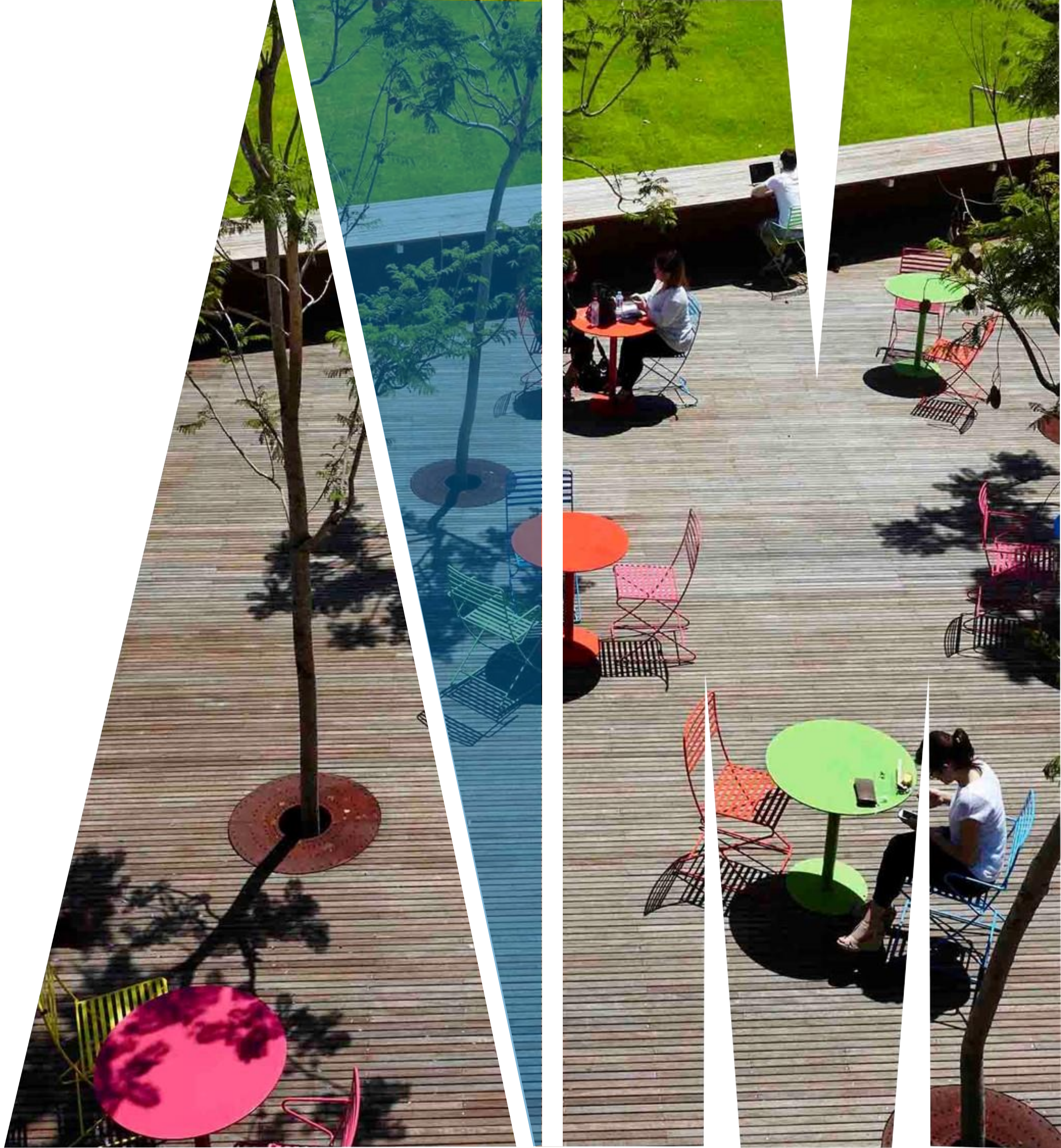
MONASH University

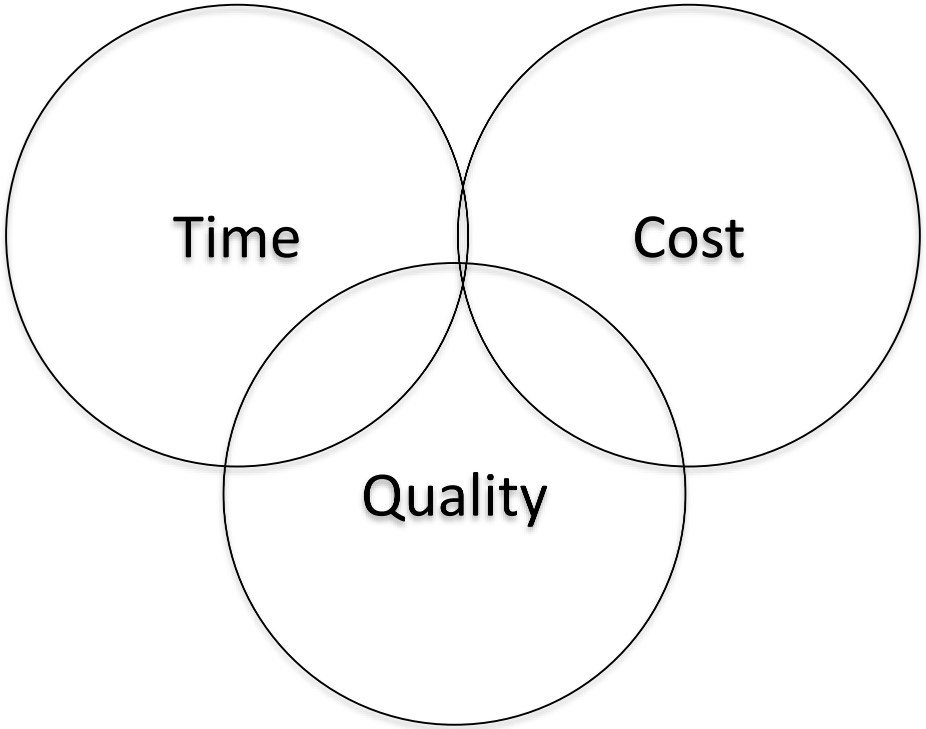
FIT2099 Object-Oriented Design and Implementation

Introduction to object-oriented design



THE
PURPOSE OF THIS UNIT

We want to be able to deliver software on **time**, on **budget**, and of sufficient **quality**



THE
PURPOSE OF THIS UNIT

Software can take a long time to develop

- small, trivial projects might take weeks but large and complex projects take years
- we want to teach you skills that will let you work on **large** and **complex projects**

"The real breakthrough came when we realized that **software is infinite**, while projects are finite. This is the approach to software development that will break the chains that hold back our advances. Make 2020 the end of software projects…"

— Standish Group Chaos Report (2020) Beyond Infinity

SO WHAT?

It’s easy to make small programs that run once

- and you can get away with many, many bad practices in doing so

We want to build software that:

- is **large** (perhaps *millions* of lines of code)
- has an acceptable number of bugs, and **few** other **quality problems**
- **can be fixed** easily when bugs are discovered
- **can be extended** or modified easily when users’ needs change

All delivered within reasonable **time** and **budget**

This is not a solved problem!

WHERE
FIT2099 FITS IN?

Design

- making good decisions about how system is put together

Our key focus in FIT2099

Quality Assurance

- checking that artefacts (code and non-code) produced in the process are of satisfactory quality

Touch on in 2099; main focus of FIT2107

Management/Process

- making these and other essential activities happen in a team, at the right time.

Touch on in 2099; main focus of FIT2101

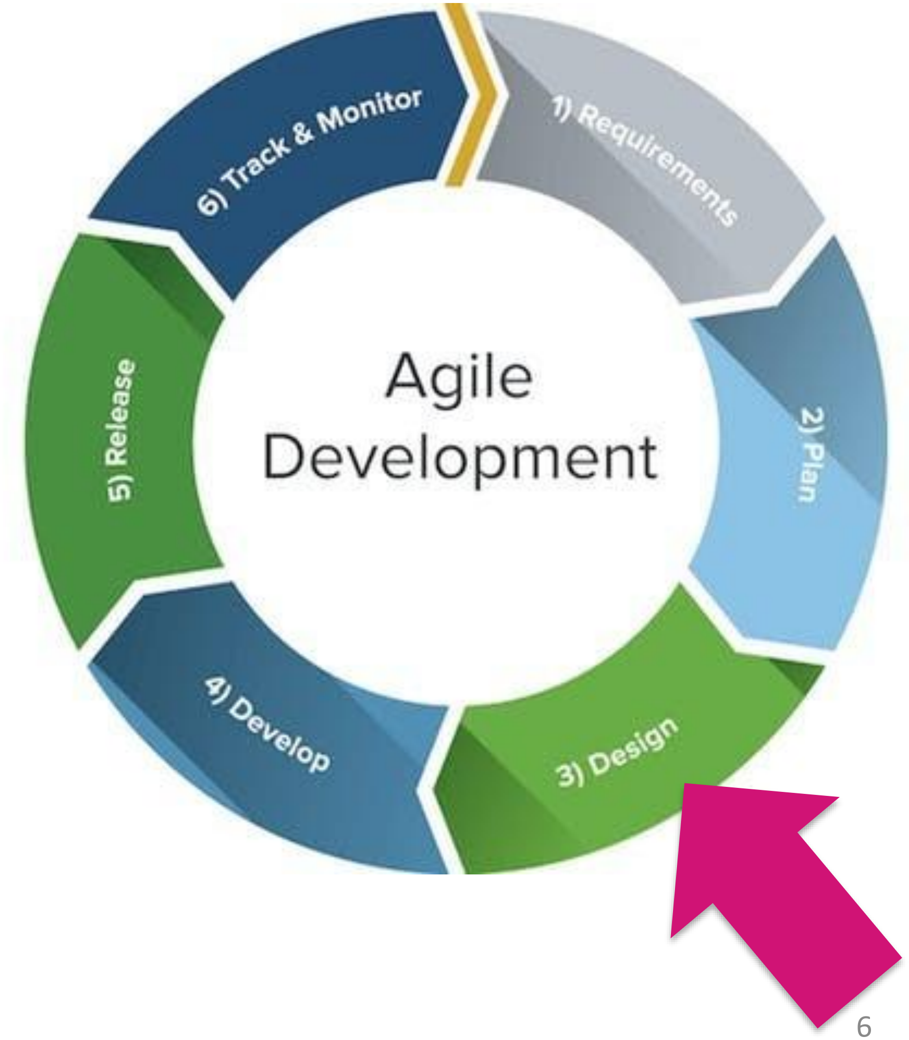
WHAT IS DESIGN?

Design is **NOT** the production of a “design document” in the “design phase”

Design **IS** actually the process of making decisions about how the software is to be implemented so as to have all the desired qualities

- of which functionality is only one
- maintainability, extensibility also very important

Design is distinct from software requirements analysis



WHY IS DESIGN IMPORTANT?

You don’t need to put much effort into design if you’re going to build something small and low-stakes, like this doghouse…

…but we want to help you develop the skills to enable you to build the software equivalent of skyscrapers and bridges – large, expensive software projects, with key requirements for performance, security, stability, and many other aspects.

WHY THEN
OBJECT ORIENTED-PROGRAMMING?

OOP Ideas developed in ‘60s and ‘70s

- Simula 67 – Ole-Juhan Dahl and Kristen Nygard
- Smalltalk – Alan Kay and others at Xerox PARC

Inspired by biological systems

Popularized by C++, then Java, C#, etc.

- Nearly all modern languages have capabilities or concepts originating in OO (Python, Javascript, Ruby, Scala, etc.)

OOP has proven good for constructing **large systems**

SOME LIMITATIONS WITH
EARLY, LOW LEVEL LANGUAGES?

Languages such as Assembler, BASIC, FORTRAN

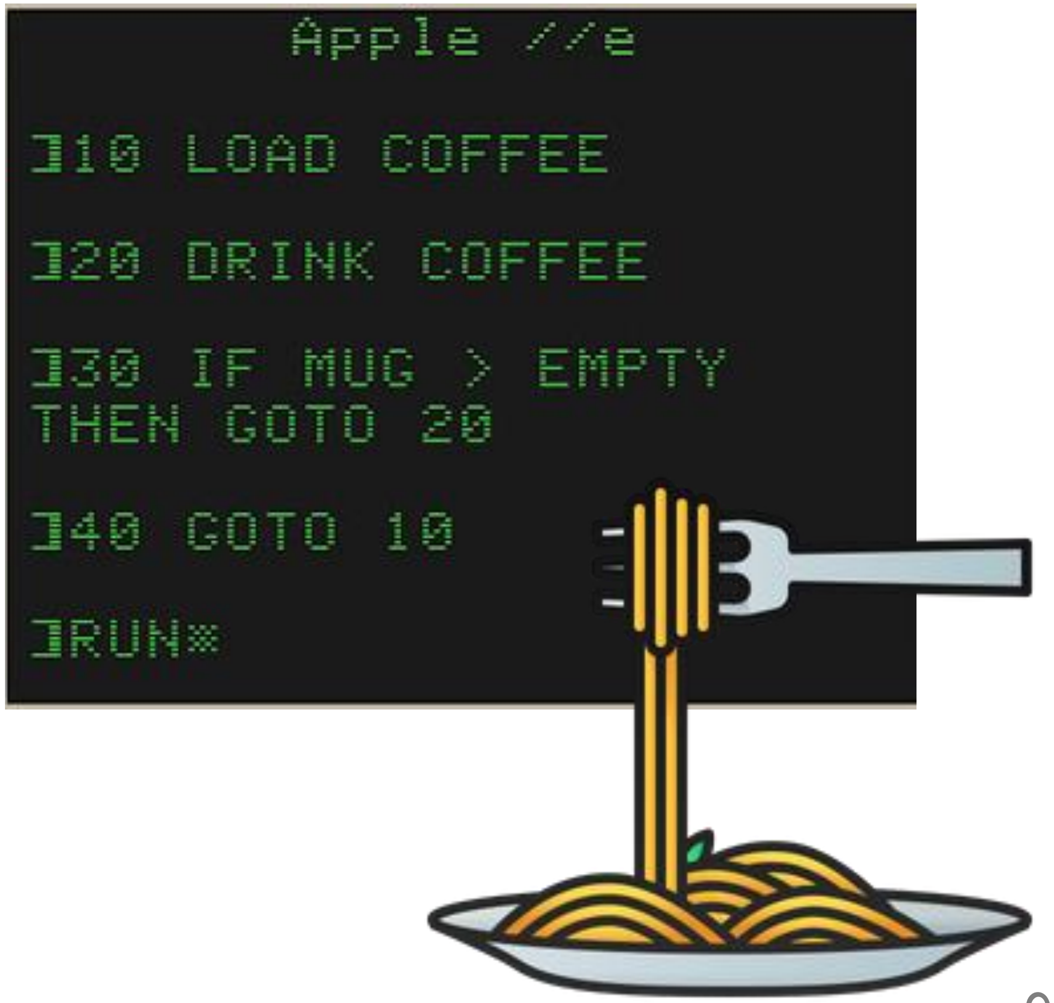
Programs were **unstructured lists of statements**.

GOTO let you jump from any statement to any other statement

All variables had **global scope**

Nightmare to **debug** and **modify**

Not always portable



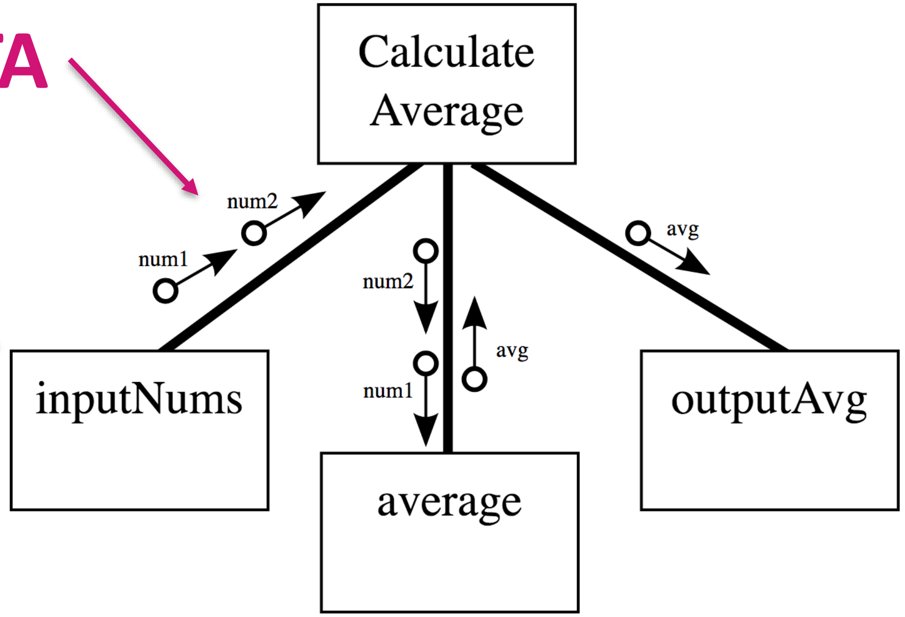
```
        Apple //e

]10 LOAD COFFEE

]20 DRINK COFFEE

]30 IF MUG > EMPTY
THEN GOTO 20

]40 GOTO 10

]RUN
```

PROGRAMMING PARADIGM:
PROCEDURAL

Programs consist of procedures (**actions**) that pass **data** to each other

DATA

CODE



Notice that the boxes represent code (procedures) and the **lines represent data**

Structure Chart Example
https://tinyurl.com/y7ysyhdk

DISADVANTAGES WITH
PROCEDURAL PROGRAMMING

Now code was easier to write but **maintenance can still a pain**

- **big systems were still hard to build**

Procedural programming makes **action** the primary unit of organization

- **data is secondary**

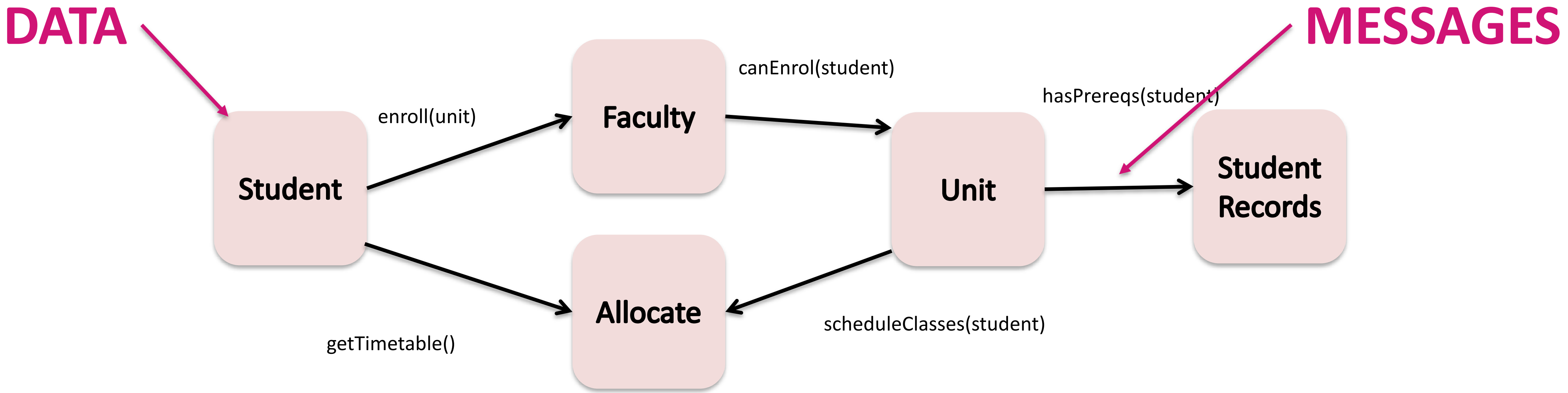
So what happens when the same data structures are needed in many places in the application?

- lots of **repeated** code
- lots of **coupling** (i.e. changes in one function or data structure means you have to change something elsewhere)

PROGRAMMING PARADIGM:
OBJECT ORIENTED

Object-oriented programming **flips this around**

Programs consist of objects (data) that send messages to each other

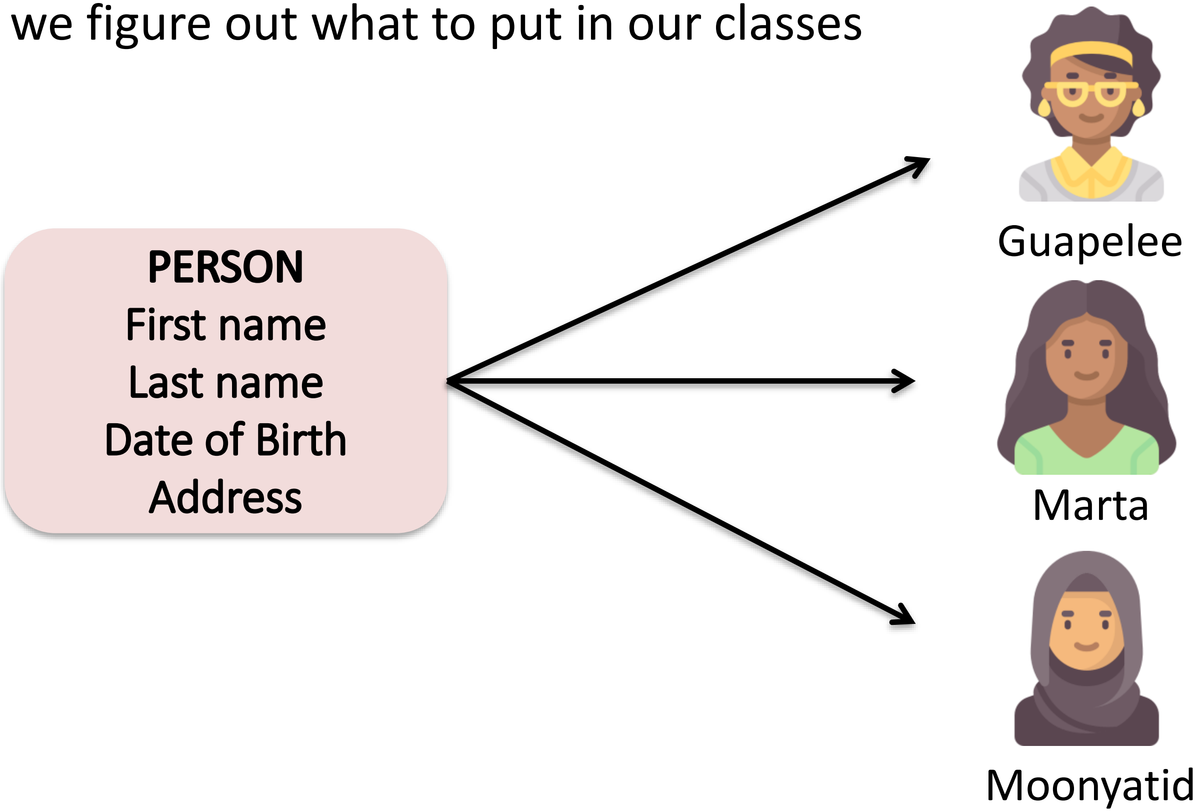


Here, the nodes represent **data** and the **arrows represent code** (method calls)

TWO KEY CONSTRUCTS BEHIND
OBJECT ORIENTED PROGRAMMING

Abstraction: figure out how to represent complex things in simple ways

- by identifying what’s important about the thing in the context of the software
- this is how we figure out what to put in our classes



PERSON
First name
Last name
Date of Birth
Address

Guapelee

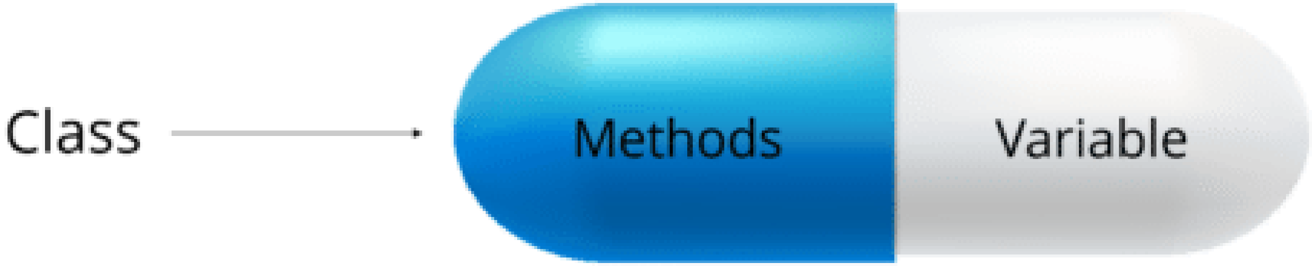
Marta

Moonyatid

TWO KEY CONSTRUCTS BEHIND
OBJECT ORIENTED PROGRAMMING

Encapsulation: bundle data that represents an abstraction together with the functions that operate on it

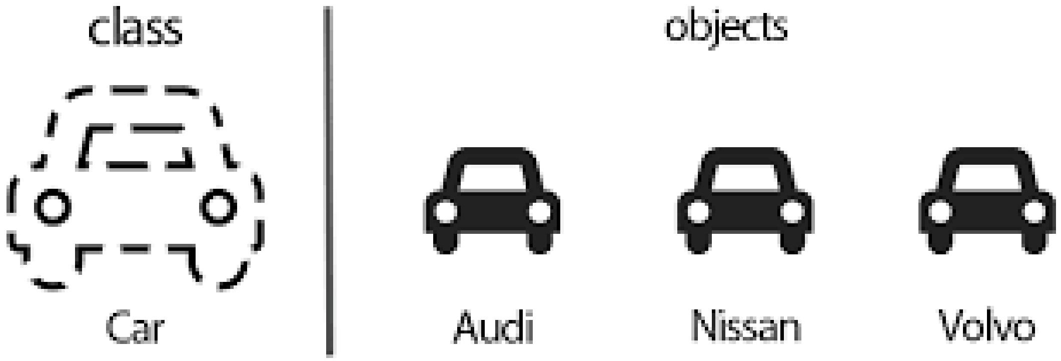
- then **hide implementation details** from other bundles (i.e. classes)
- so you **don’t need to think about the implementation**



THE
CLASS

A class is an extensible program-code-template for creating objects, providing:

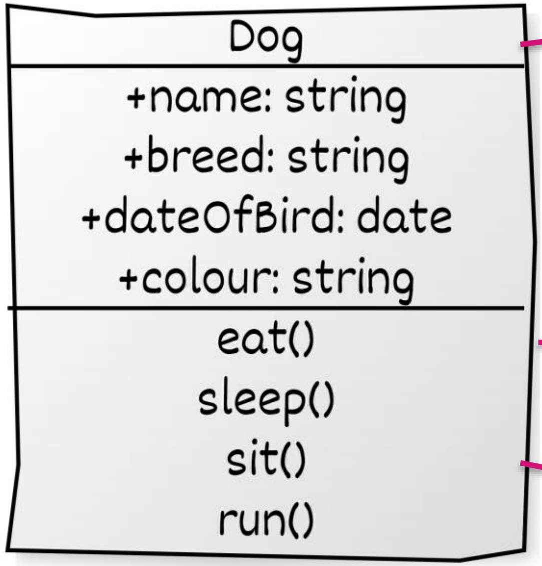
i) initial values for state (attributes) and
ii) implementations of behaviour (member functions or methods)



THE CLASS AS AN ABSTRACT DATA TYPE

Attributes
(data)

Methods
(behaviour)



Dog

+name: string
+breed: string
+dateOfBird: date
+colour: string

eat()
sleep()
sit()
run()

Name: Buddy
Breed: Boxer
Age: 1 month
Colour: white/brown

Name: Buddy
Breed: Luna
Age: 1 year
Colour: black/white

Name: Dorito
Breed: Chihuahua
Age: 3 years
Colour: light brown

THE OBJECTS

A class is a definition that says what data and methods get bundled together

- e.g. “A Patient has a name, an attending doctor, and a diagnosis, and has a diagnose() method and a bill() method”

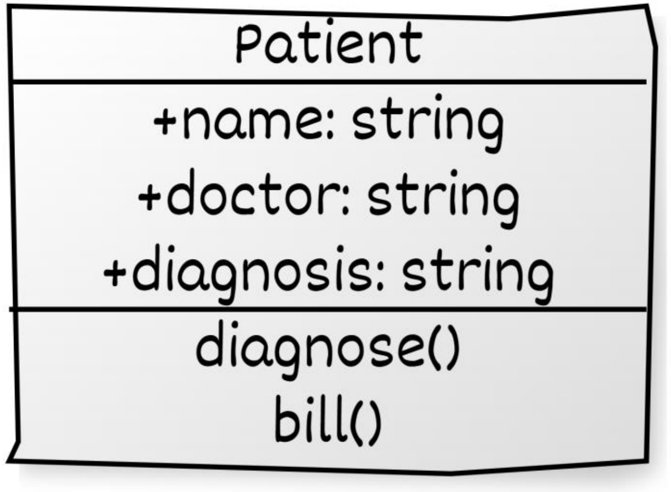
An object is an instance of a class

- that is, it’s one specific collection of data
- e.g. “Ravi, Dr Chang, broken leg”
- in Java, all instances of a class share the same method code
  - in JavaScript, you can assign new functions to objects, but you can’t do this in Java

THE MESSAGES (METHODS)

You send a message to an object by **calling one of its methods**

- that is, the collection of methods within a class define which messages its instances can respond to



Patient

+name: string
+doctor: string
+diagnosis: string

diagnose()
bill()

For example:

```
Patient ravi;    // Ravi is a patient
…
ravi.diagnose(“common cold”);
ravi.bill();
```

Here, we’re creating a Patient called ravi and sending it a “**diagnose**” message and a “**bill**” message

Nearly all of you will have seen this in other units, but you might not have thought of it as **messaging**

BENEFITS
OF OOP

A key design principle:

Reduce Dependencies as much as possible

- you will hear it again and again in this unit and others

The corollary is:

If things *must* depend on each other, group them together
(inside an encapsulation boundary – more on that later)

- you will hear this one a lot too

Actions that access or modify a certain data item must depend on that data, and on each other

- so it makes sense to group them together in a class
- this makes it easier to limit the scope of changes to a single class

WHY JAVA?

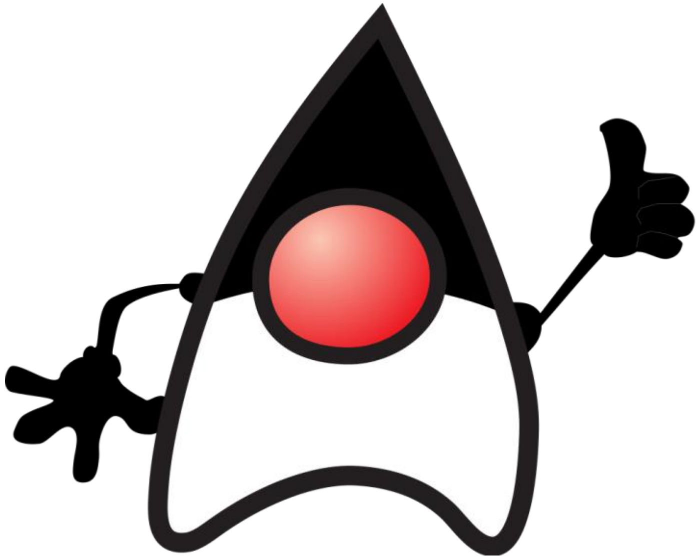
As a teaching language:

- fairly pure OO language
  - keywords match very well with the key OO concepts
    - `extends`, `implements`, `public`, `private`, etc.
- strong, static typing
- memory safety (garbage collection)
- widely used in industry
- free tools available on all major platforms (MS Windows, MacOS, Linux,…)

As an industry language:

- widely used for enterprise systems
- supports large systems well
- high importance of reliability, maintainability, security

Default language for Android development



Duke, the Java mascot

In fact, Java is one of the top-most used programming languages:

https://statisticstimes.com/tech/top-computer-languages.php

WHY NOT JAVA?

Verbose

- partly inherent to being an industrial-scale OO language
- partly due to design
- tedious to write without IDE support

Harder to use **platform-specific toolkits** (particularly UI)

- if you want to target Windows, C# is easier
- if you want to target MacOS, Objective-C or Swift are easier
- if you want to target browser front-ends, Javascript is easier

Supplied class **libraries not newbie-friendly**

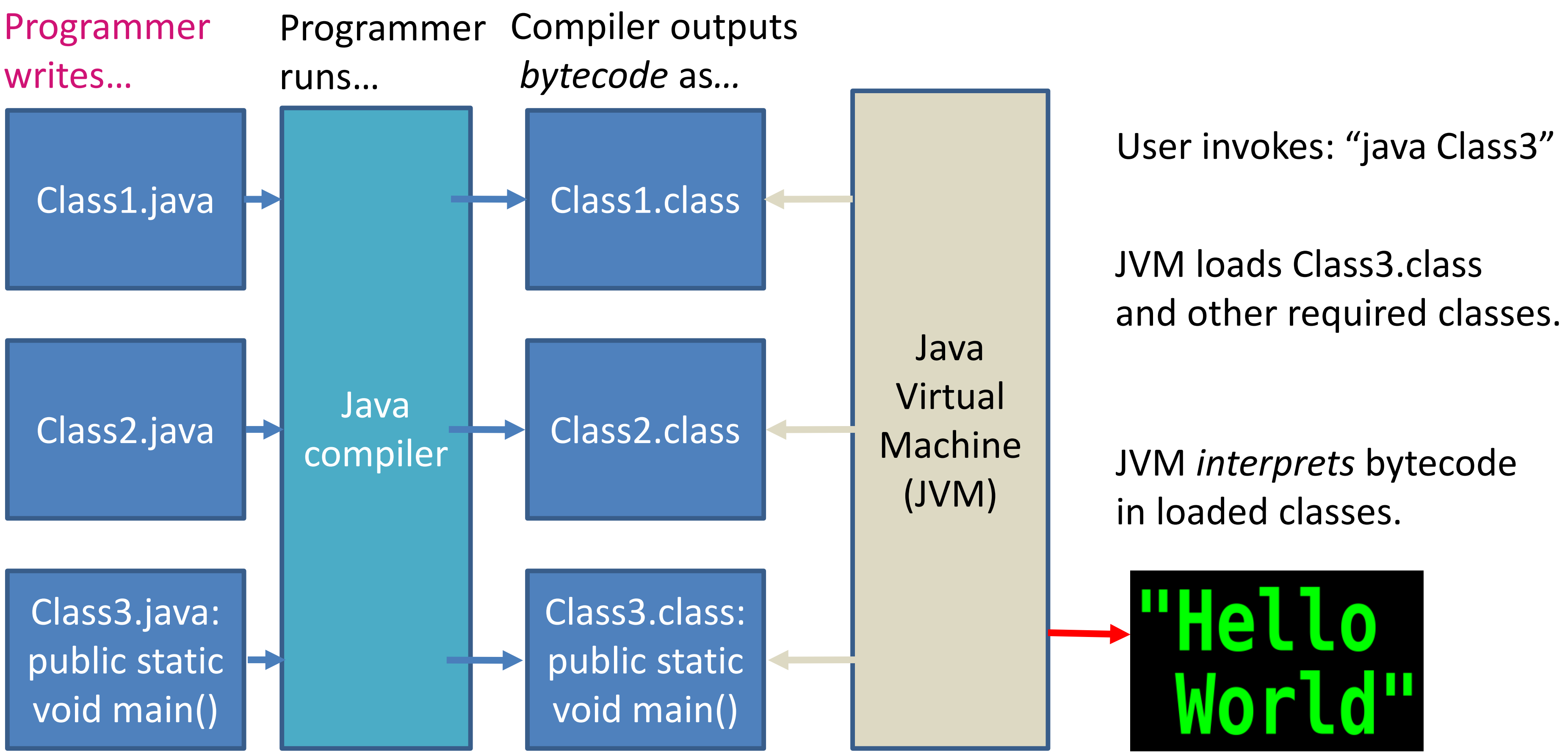
Slower than C/C++ in some circumstance

ALTERNATIVE LANGUAGES

Learning JAVA concepts will help you learn other popular languages

- C# is ***very* similar** to Java
- the concepts you will learn in FIT2099 will also help you learn OO features of **C++, Swift, Python, Ruby**, and many other languages
- the class model of **JavaScript** is very different, but **message passing** works in a similar way

HOW YOUR
JAVA PROGRAM WORKS?



Programmer writes…

Programmer runs…

Compiler outputs *bytecode* as…

Class1.java

Class2.java

Class3.java: public static void main()

Java compiler

Class1.class

Class2.class

Class3.class: public static void main()

Java Virtual Machine (JVM)

User invokes: “java Class3”

JVM loads Class3.class and other required classes.

JVM *interprets* bytecode in loaded classes.

"Hello World"

THE
INTELLIJ IDE

It’s possible to

- write Java programs in a text editor
- compile by running the compiler at the command line
- run by invoking the JVM

But this can be tedious

Integrated Development Environments (IDEs) take most of the tedium out

Several IDEs for Java in wide use:

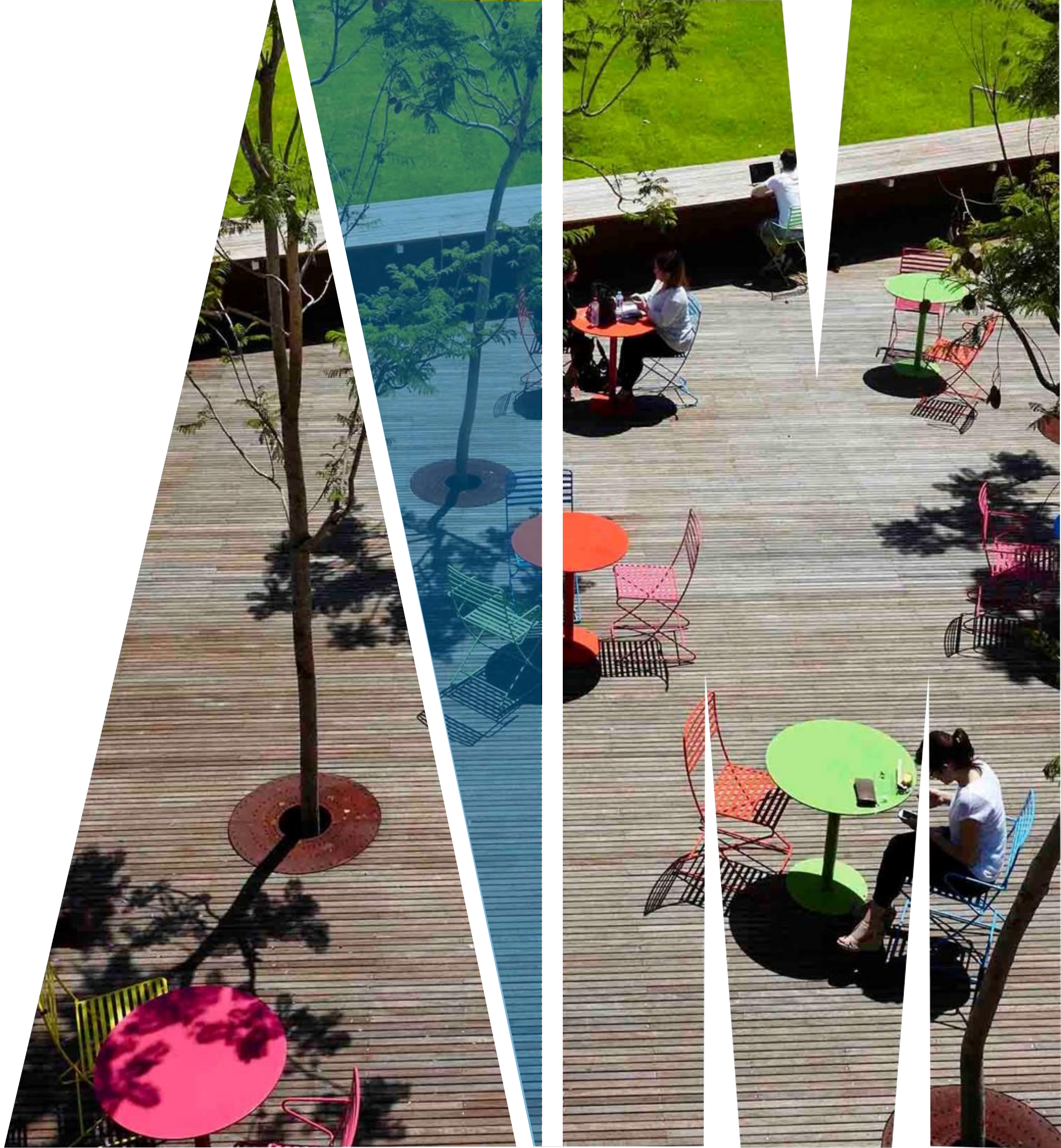
- IntelliJ IDEA https://www.jetbrains.com/idea/
- Eclipse https://www.eclipse.org/ide/
- Netbeans https://netbeans.org/

You can use any tools you like for FIT2099

We will be using and supporting the IntelliJ IDE

MONASH University

Thanks



MONASH University