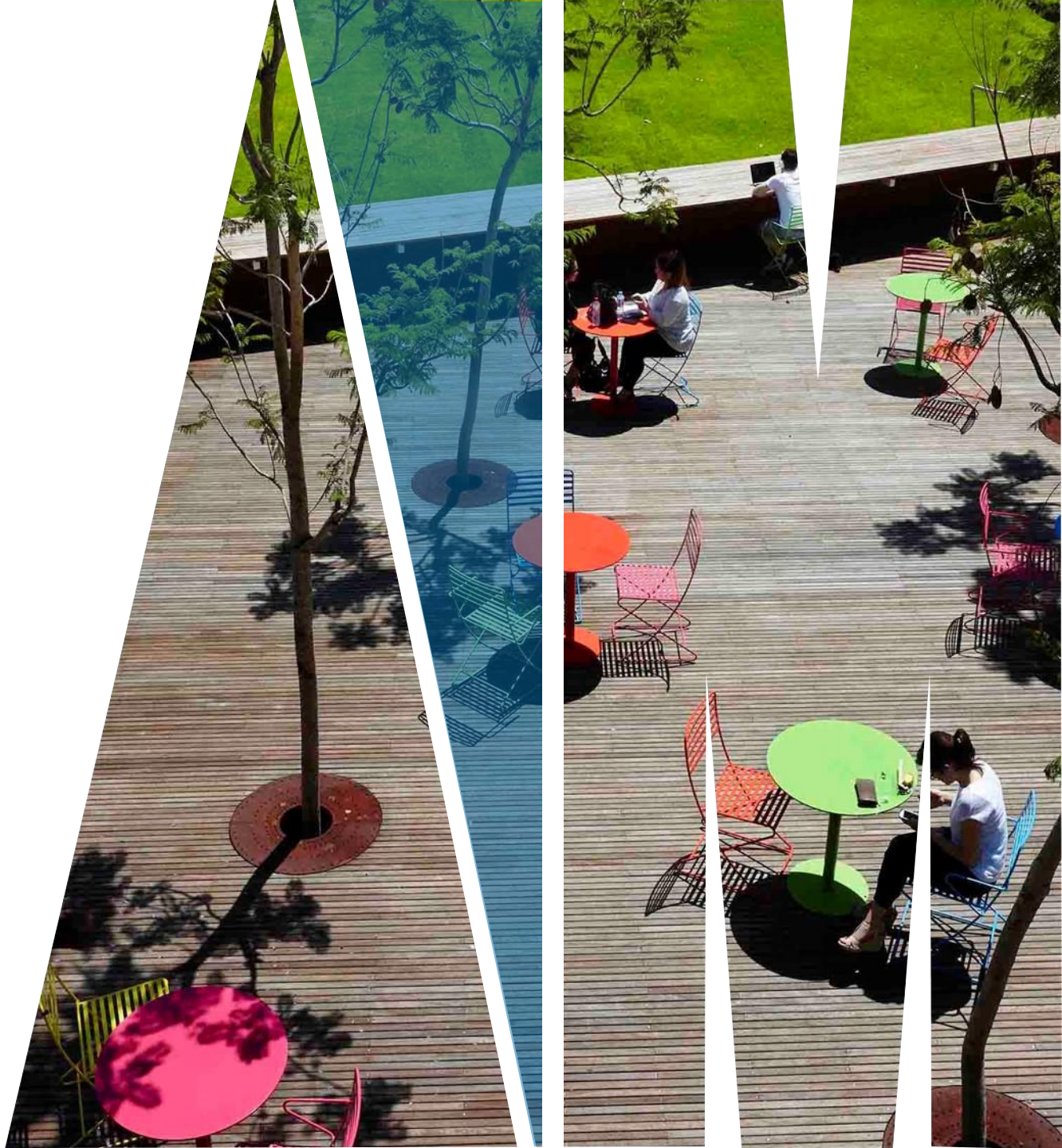
MONASH University

FIT2099 Object-Oriented Design and Implementation

Generics

(abstraction)



MONASH University

WHY GENERICS?

Sometimes Java's strongly typed nature gets in the way of good use of abstraction

- example: suppose you want to write a program that requires a set of integers and a set of strings
- the set operations and logic behind these sets would be the same, so we would like to be able to define a **Set class** to generalise these
- but we'd need to be able to store **both Integers and Strings** in it…
- would also be nice to be able to re-use this Set code in other projects

THE GENERICS IN JAVA

Java addresses this problem using **generics**

- you have seen these in the Collection framework (e.g. **`ArrayList<>`**)
- essentially, we supply the name of the class as a **parameter** inside **angle brackets**
- let's take a closer look at how they developed and how they work

C# also uses Java-style generics. **C++** uses templates.

Python's duck typing means that it doesn't need such a mechanism, although it also means that it's not possible to constrain the relationships between classes.

WHAT ARE PARAMETERISED TYPES?

Generics means **parameterised types**.

The idea is to allow type (Integer, String, … etc, and user-defined types) to be a parameter to methods, classes, and interfaces.

Using Generics, it is possible **to create classes that work with different data types**.

An entity such as class, interface, or method that operates on a parameterized type is called **generic entity**.

WHAT IS A
GENERIC CLASS

Like C++, we use <> to specify parameter types in generic class creation. To create objects of generic class, we use following syntax.

// To create an instance of generic class
BaseType <Type> obj = new BaseType <Type>()

*Note: In Parameter type **we cannot use primitives like 'int','char' or 'double'.***

ADVANTAGES OF GENERICS

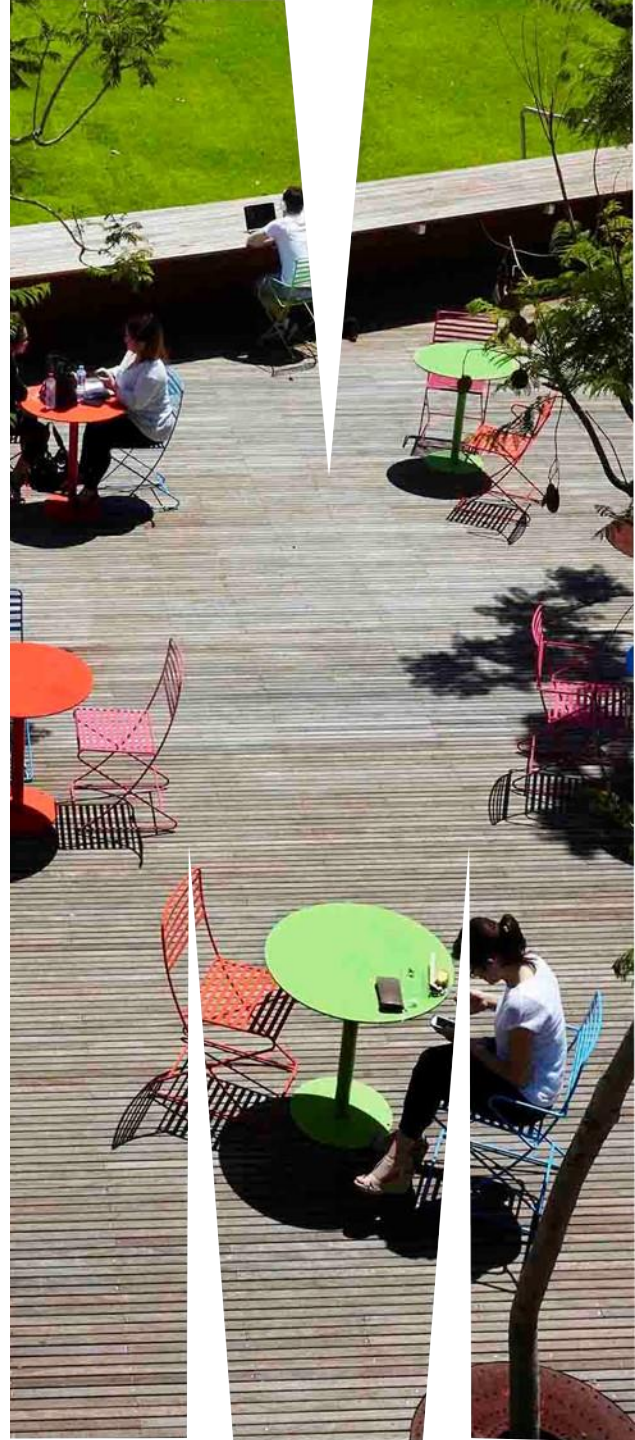
Programs that uses Generics has got many benefits over non-generic code.

Code Reuse: We can write a method/class/interface once and use for any type we want.

Type Safety: Generics make errors to appear **at compile time** rather than at in run time (it's always better to know problems in your code at compile time rather than making your code fail at run time).

MONASH University

Thanks



MONASH University