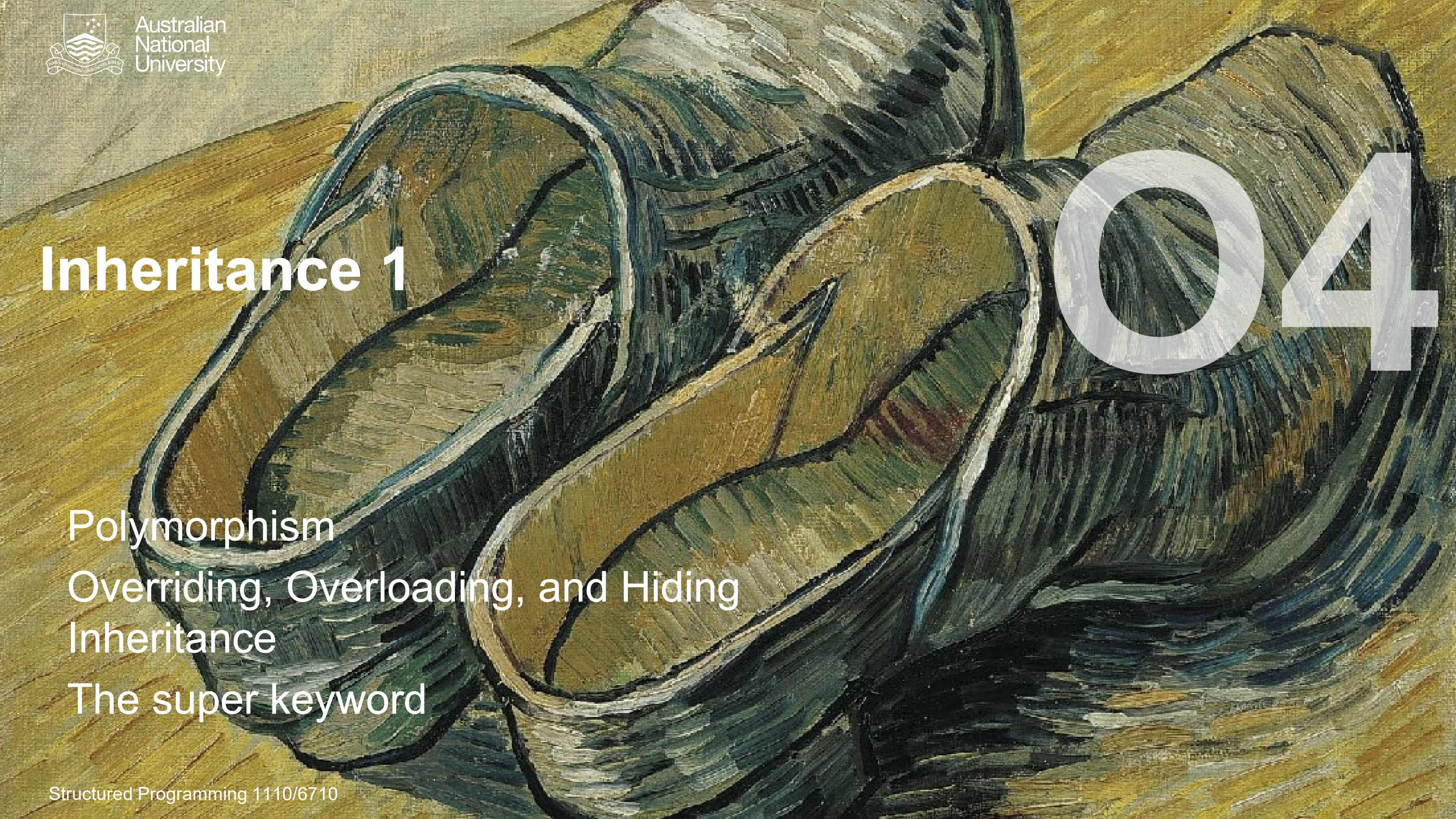
Australian National University



Inheritance 1

04

Polymorphism

Overriding, Overloading, and Hiding

Inheritance

The super keyword

(Subtype) Polymorphism

A reference variable may refer to an instance that has a more specific type than the variable.

The code that is executed on an instance method call depends on the type of the instance, not in the type of the variable.

Inheritance and Subtyping

A class that `extends` another class is known as a *subclass*, *derived class*, or *child class*. The extended class is known as a *superclass*, *base class*, or *parent class* (similar for extended/implemented interfaces).

- A subclass inherits the members of its superclass
  - Can override/hide their definitions
- Each class `extends` exactly one other class (*single inheritance*)
  - All classes implicitly inherit from `java.lang.Object`
- Each class/interface can implement/extend arbitrarily many interfaces (*multiple inheritance*), but no code or field inheritance

Inheritance and Subtyping

Each class/interface type is treated as a *subtype* of the types of the classes/interfaces it extends/implements.

Liskov Substitution Principle / Behavioral Subtyping

A type `S` is a subtype of a type `T` if every instance of `T` could be substituted by some instance of `S`.

➔ Subtyping is about behavior; inheritance is about code reuse.
Java conflates the two a bit.

Overriding	Hiding		Overloading	
Instance Method	Class Method	Instance/ Class Field	Instance Method	Class Method
Same signature …		Same name …	Same name, different parameters …	
… as instance/class method/field in superclass			… as any class/instance method in current or superclass	
Type of instance determines the method	Type of variable (instance) or relevant class determines method/field			

☺ ☹ 😐

The `super` keyword

You can access overridden (or hidden) members of a superclass by using the **`super`** keyword to explicitly refer to the superclass.

Key use cases:

- Calling superclass constructors by using `super()` with appropriate arguments
- Calling implementations of overridden methods to extend their behavior, using `super.`[methodName]`()` with appropriate arguments