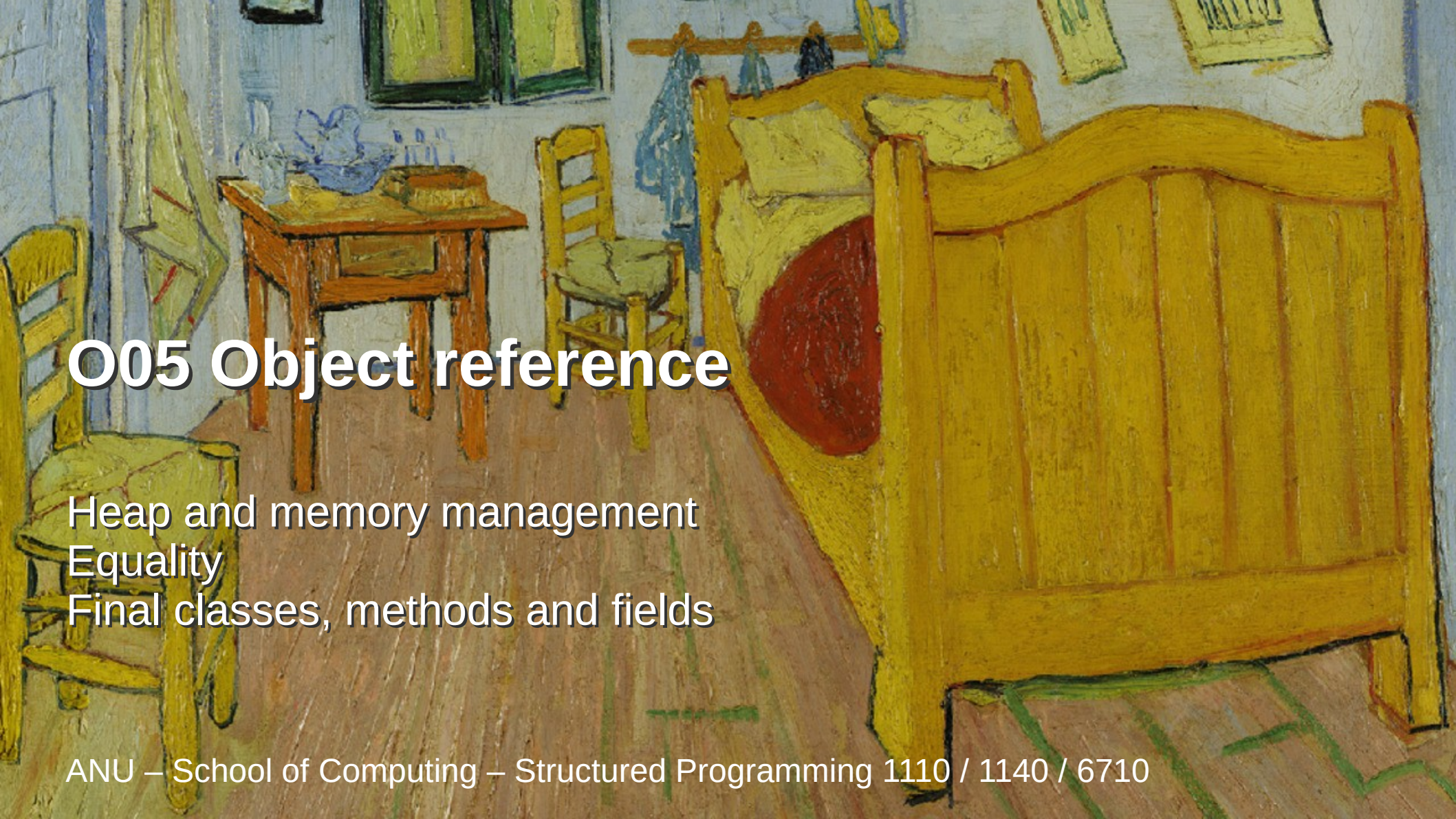
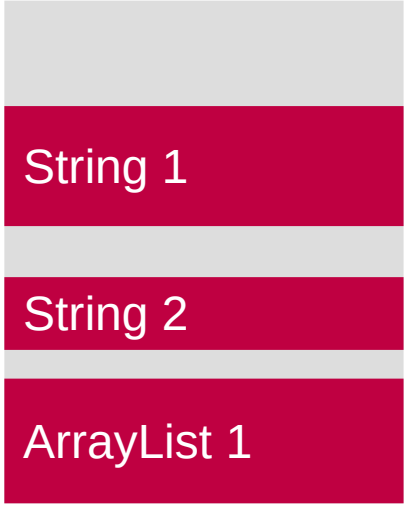


O05 Object reference

Heap and memory management
Equality
Final classes, methods and fields

The Heap

The heap: a large region(s) of memory used to store dynamically allocated objects (objects created with `new`).



Variables and References

- For variables of **primitive types**, the value is stored directly.
- For variables of **reference types (all objects)**, the “value” stored is a *reference* to an object stored on the heap.
  - Such variables can be set to `null` (reference to nothing).
  - Method calls, fields automatically access the object pointed to.
    - `NullPointerException` thrown if reference is `null`
  - More than one variable can *refer to the same object*.

Equality

- Variables of **primitive types**:
  - Use == for equality.
  - Have no methods (i.e. have no `equals()`).
- Variables that **reference objects**:
  - `a == b`: true iff a and b refer to the **same object instance**.
    - Checking the variable's immediate value is the same, which is a reference.
    - Two different instances can have exactly the same fields, and yet not be ==.
  - `a.equals(b)`: class-specific (semantic) object equality.
    - Default inherited from `java.lang.Object` is just ==.

Garbage Collection

In Java, there is no explicit deallocation of objects.

A *garbage collector* automatically reclaims heap space used by objects that are no longer reachable (no longer referenced, directly or indirectly, by any variable in the program).

The `final` modifier

- A `final` field can not be reassigned
- A `final` method cannot be overidden
- A `final` class cannot be subclassed.

A `static final` field of a primitive type is like a constant.

A `static final` field of a reference type will always refer to the same object, but that object may change.