Lesson Notes: 04/02/2026

The 'Contract' for Overriding equals()

Look at the documentation on the equals() method and show the 'contract' for equivalence between objects.
To paraphrase, an object implementing equals() should return true if:

- The comparison isn't with a null object reference, it returns false if it is
- It is a comparison with itself (x.equals(x)) (Reflexive)
- The other object would also return true (x.equals(y) -> y.equals(x)) (Symmetric)
- There is a shared equivalence with a third object (x.equals(y) -> y.equals(z) -> z.equals(x)) (Transitive)
- It has previously return true as long as the object states remain unchanged (Consistent)

What is a Hash Code?

A hash code is an integer that is generated from a given input using a hashing algorithm. It has the following characteristics:

- It is a one-way process, it is practically impossible to reverse the algorithm to get the original input
- The integer has a fixed size and is therefore independent of the size of the input
- The same input will always produce the same hash code
- There is a slight chance that two distinct inputs can produce the same hash code

The hashCode() Contract

The 'contract' for overriding the hashCode() method is what ensures this consistency. In short:

- During the running of a program, as long as the information used in the equals() comparison doesn't change, the integer result of hashCode() shouldn't change either
- If two objects are equivalent according to equals(), hashCode() should return the same integer for both
- Two objects that are not equivalent (by the same measure) may also have the same integer result returned by hashCode() - but this degrades the performance of hash tables.