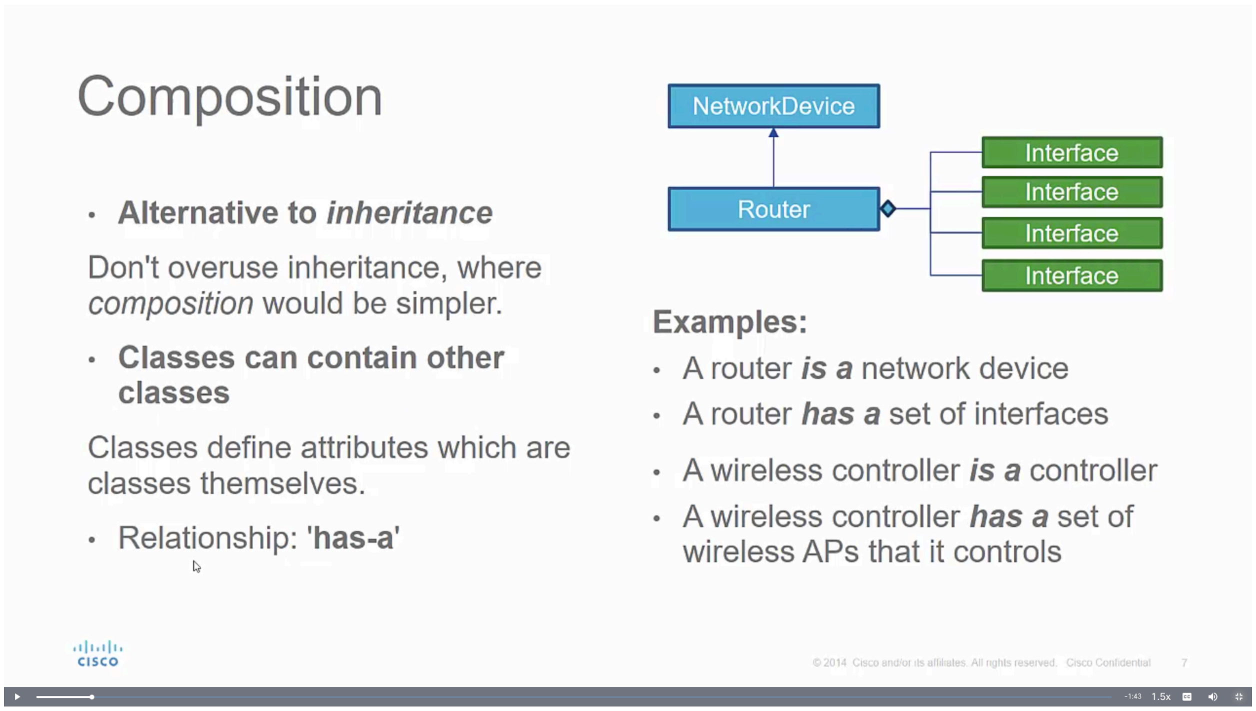


Composition

- **Alternative to *inheritance***

Don't overuse inheritance, where *composition* would be simpler.

- **Classes can contain other classes**

Classes define attributes which are classes themselves.

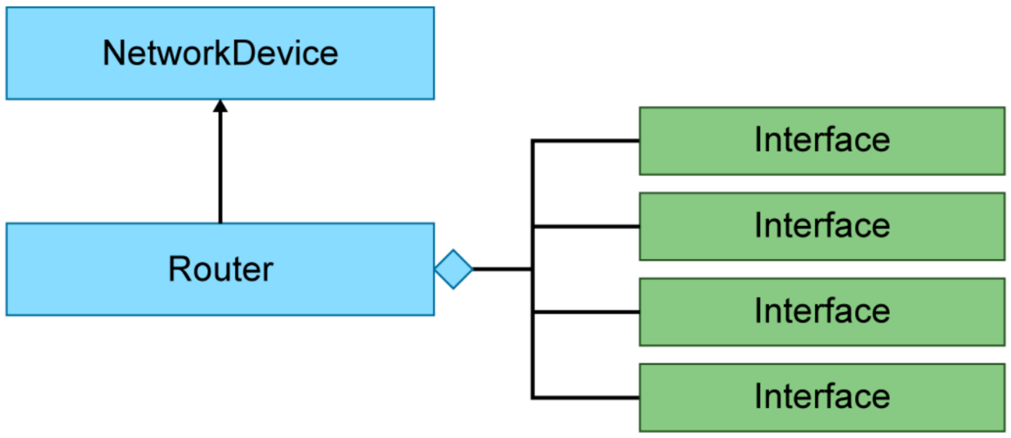
- Relationship: '**has-a**'

Examples:

- A router ***is a*** network device
- A router ***has a*** set of interfaces
- A wireless controller ***is a*** controller
- A wireless controller ***has a*** set of wireless APs that it controls

Inheritance is a powerful feature of OOP, but overuse of this feature can be problematic. Therefore, before embarking on the user of inheritance in your classes, consider whether composition may be a preferable design for your classes.

As mentioned, inheritance represents an is-a relationship. An IOS-XR router is a network device. Composition, on the other hand, represents a has-a relationship. An IOS-XR router has a set of interfaces. An IOS-XR router has a set of static routes.



These examples are fairly straightforward. Some examples that may require careful thought might be the question if an XR router has an OSPF state? Or is it an OSPF router?

Helping to understand the question may be whether your device would have to inherit from multiple parent classes, for example, if a router is both an OSPF router and an IPv4 router. Inheriting from multiple parent classes is allowed in Python, but quite often it can lead to confusing and complicated code, and thus should be considered with care.