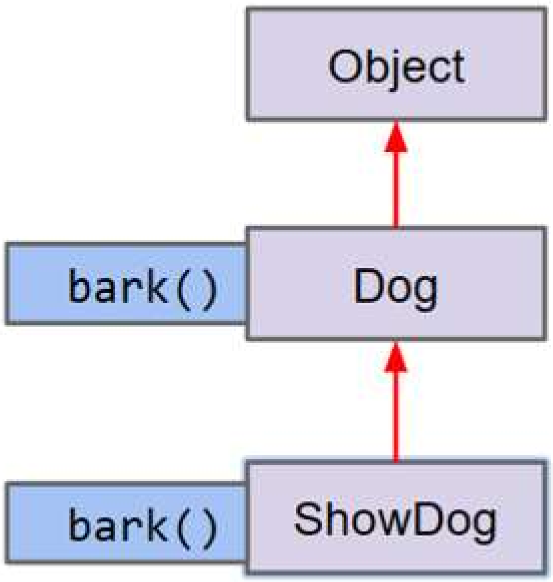
11.1 A Review of Dynamic Method Selection

In previous lectures we have gone over classes extending other classes, and to make sense of this we considered whether the sub class has an "is-a relationship" with the superclass. For example, if we had the two classes:

- Dog: Implements bark() method
- ShowDog: Extends Dog, overrides bark method

This would be a valid extension as a ShowDog is-a Dog and a Dog is-an Object. These relationships satisfy the condition of a subclass being-an instance of the super class.



With this in Mind, what are the Rules to Decide if the Code Would Run or Run into Errors?

This is a particularly tricky problem to solve but the rules to look out for include:

- Compliers will allow Memory Boxes to hold any subtype of itself
  -

    - For example, compliers will allow the Dog memory box to hold a ShowDog object as a ShowDog is a subtype of the Dog Class
- Compliers will allow calls based on Static type.
    - For example, if a variable were declared as a Dog it's static type would be a Dog and the complier would allowed it to call bark()
- **Overridden non-static methods are selected at run time based on dynamic type.**
    - **Everything else is based on static types,** inculding overloaded methods.

Static Type vs. Dynamic Type

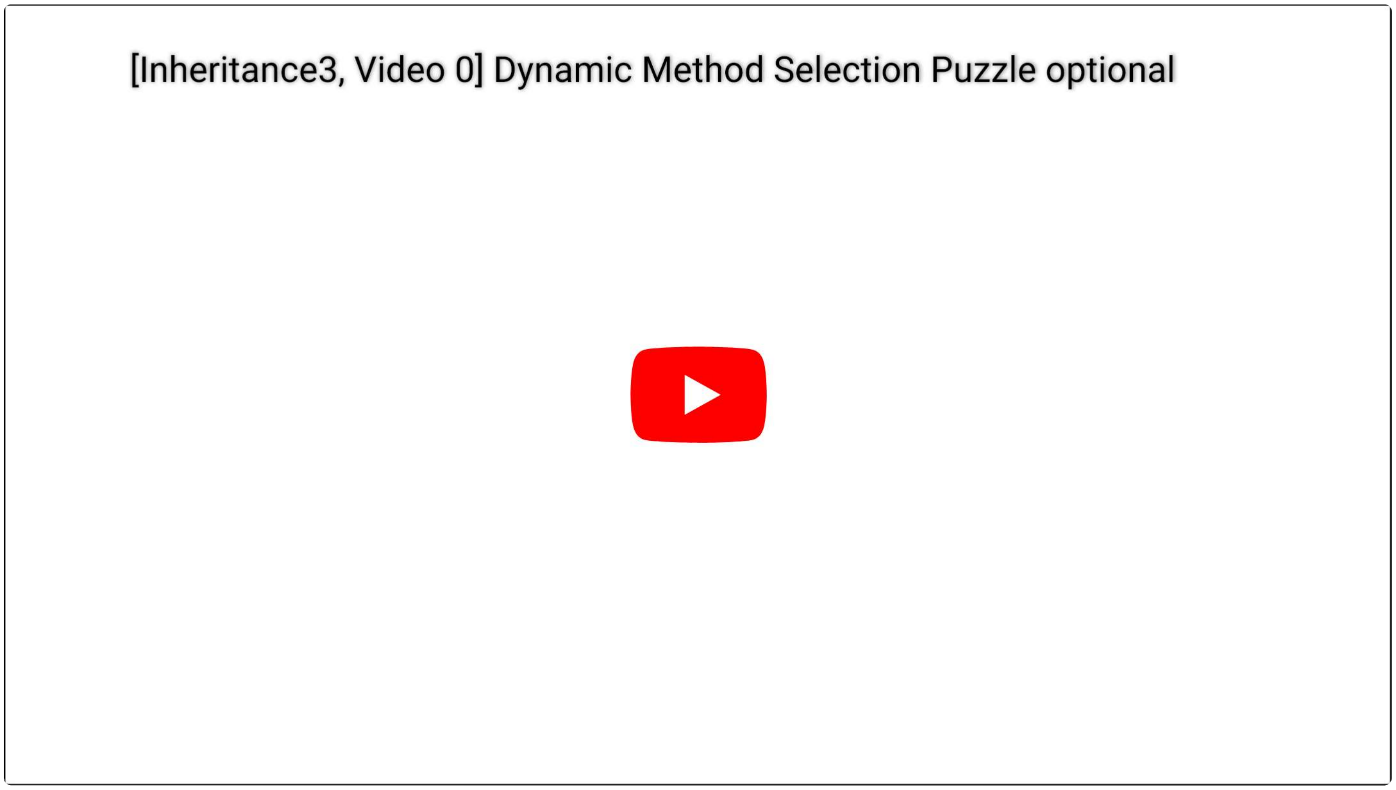
Every variable in Java has a “compile-time type”, a.k.a. “static type”.

- This is the type specified at declaration. Never changes!

Variables also have a “run-time type”, a.k.a. “dynamic type”.

- This is the type specified at instantiation (e.g. when using new).
- Equal to the type of the object being pointed at.

Accompanying Lecture


[Inheritance3, Video 0] Dynamic Method Selection Puzzle optional

Previous
11. Inheritance III: Subtype Polymorphism, Comparators, Comparable

Next
11.2 Subtype Polymorphism vs Explicit Higher Order Functions

Last updated 1 year ago