Polymorphism

(OOP Core Concept 3 - Polymorphism)

ACTLEA
Active Learning

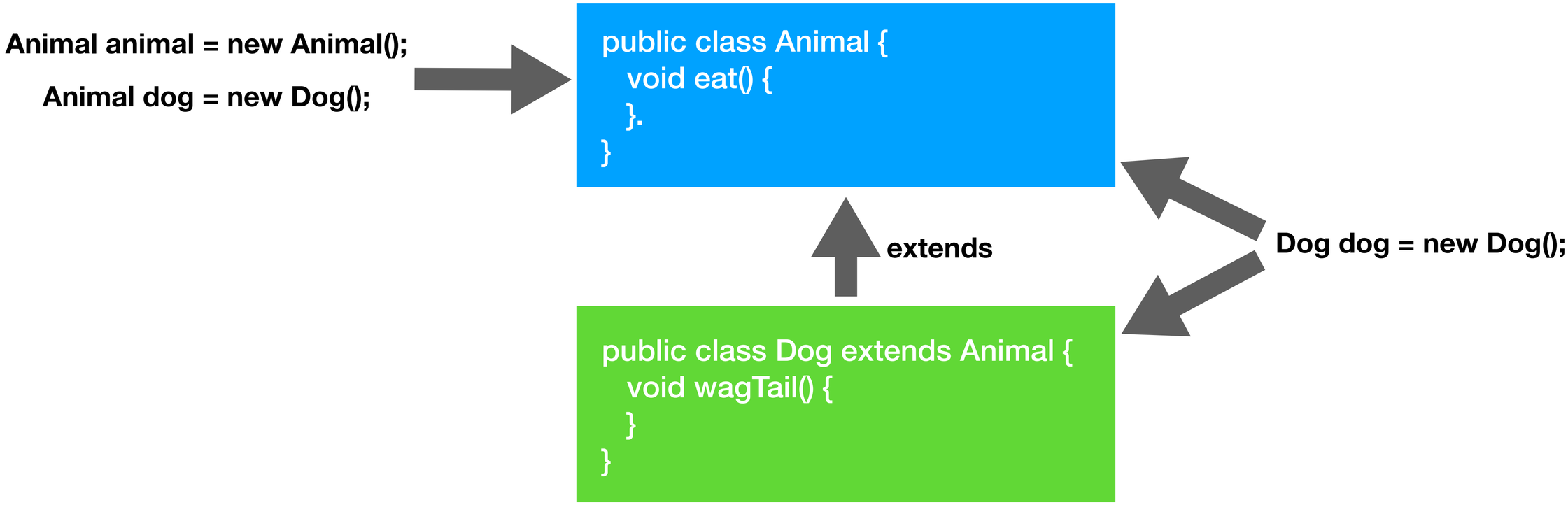
Learning objectives

- Polymorphism
- instanceof
- Overriding methods in subclasses to facilitate polymorphism

Polymorphism

- Polymorphism means the ability of an object to take on many forms
- For example a Superclass reference is used to refer to a Subclass object
- In that example the reference can only access methods and variables in the Superclass
- A Subclass object references by a Superclass reference can be cast to a Subclass reference and then everything in the Subclass is accessible

The extends keyword



Why use polymorphism?

- Handling objects on a more general level can be convenient
- (even if more specific functionality from the subclass is not available)
- For example maybe we want to make all Animals eat regardless of what type they really are

Demo 1 - Polymorphism

- Handling subclasses in a more general way by using a supclass reference

Exercise 1 - Polymorphism

- Continue on the exercise from the Inheritance module
- Create an array of type Product and assign some different subclasses to the elements in the array
- It is possible to add an object of type Book to an array of type Product since the Book inherits from Product (the Book is-a Product)
- Loop over the array and print something out from the objects (that are handled as type Product so only instance variables in Product are available)

instanceof

- instanceof is an operator find out the type of an object
- This operator is often used to find out the ”real” class of a subclass that is being referred to by a superclass type
- When the type is found, a cast can be made on the object to the real subclass type without errors, and then all the variables and methods in the subclass are available in the object

Demo 2 - instanceof

- Using instanceof to find out the type of a subclass
- Then cast the object with the superclass reference to a subclass reference

Exercise 2 - instanceof

- Continue on the previous exercise
- Create a description method in the Book class and also in the Movie class that prints a description of the object (printing some of the instance variables in the object)
- In the loop with the Product array, try to call the description method in Book objects and Movie objects. This shouldn't work since the type of the objects are Product (assuming there is no description method in Product)
- To be able to know weather an object is of type Book or Movie, use an if statement to check the type using the instanceof operator, when you know the type cast the Product object to the correct type, and if it's a Book or Movie then call the description method

Demo 2 - Override to help polymorphism

- Using override in the subclass to be able to have a more specifik implementation in the subclass that automatically runs when calling the overridden method in the superclass

Exercise 3 - Polymorphism

- Continue on the previous exercise
- Create a description method also in the Product class that prints a description of the Product (the values of the instance variables)
- Now you can call the description method on all Product objects in the loop, no need to check the type with instanceof anymore, just call description without the if statements
- If you have subclasses with more specific implementations of the description method, the more specific methods will be called automatically!

Learning objectives

- Polymorphism
- instanceof
- Overriding methods in subclasses to facilitate polymorphism