Abstraction

* What attributes and behavior

Encapsulation

* How group together and accessed

Decomposition

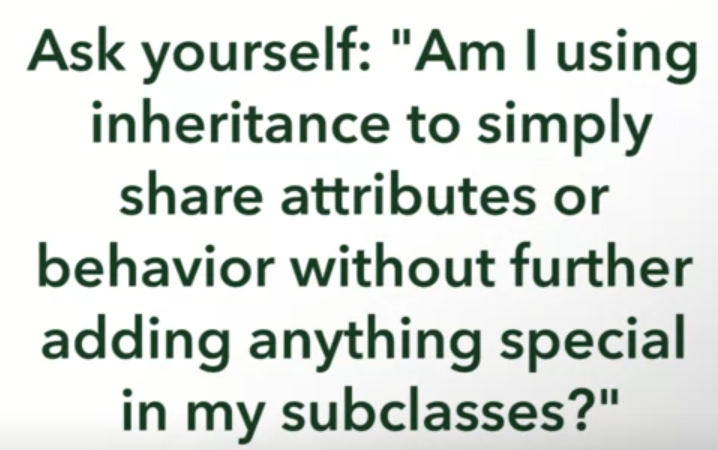
* Can my classes be simplified into smaller parts?

Generalization

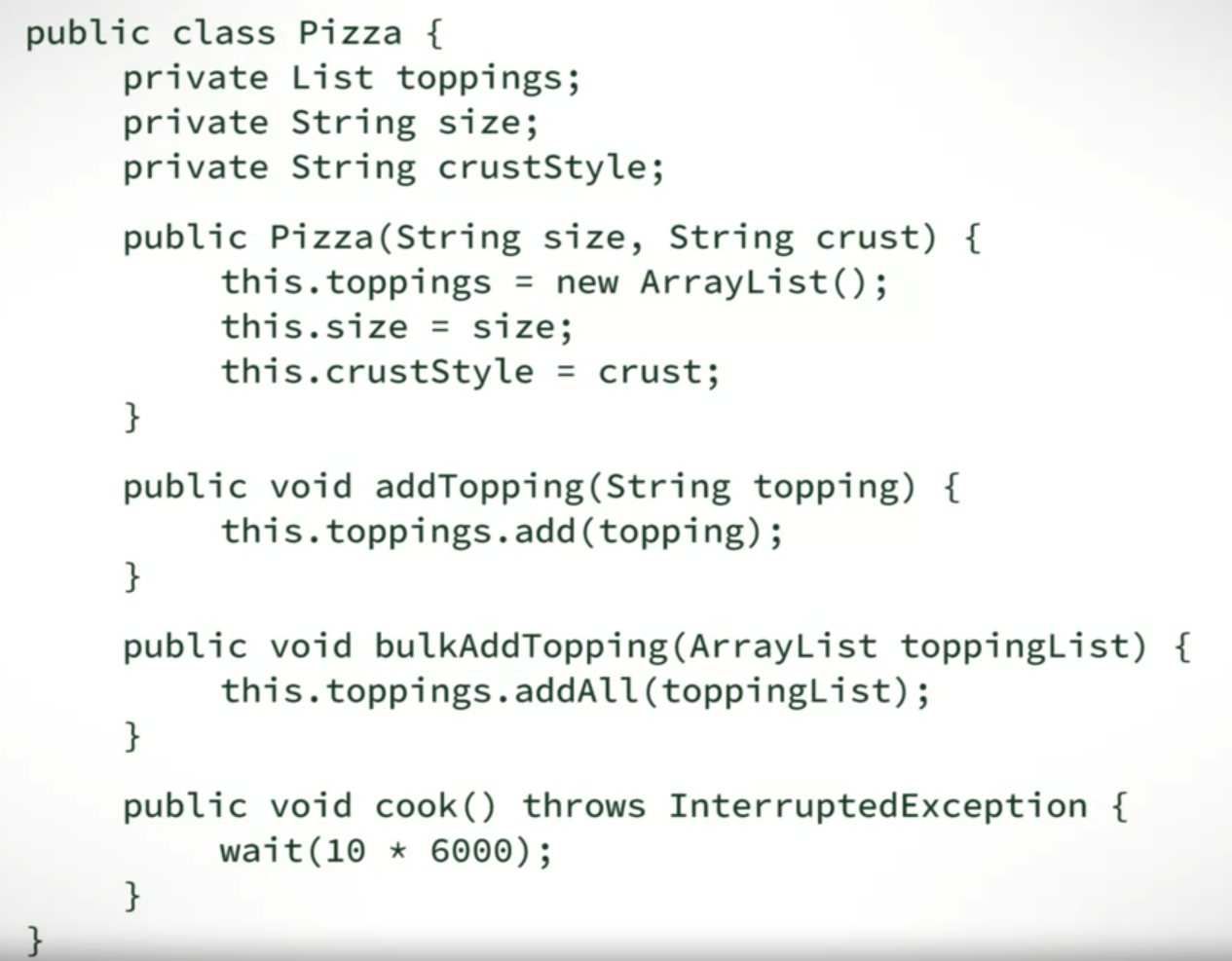
* Are there common things across my objects to be generalized?

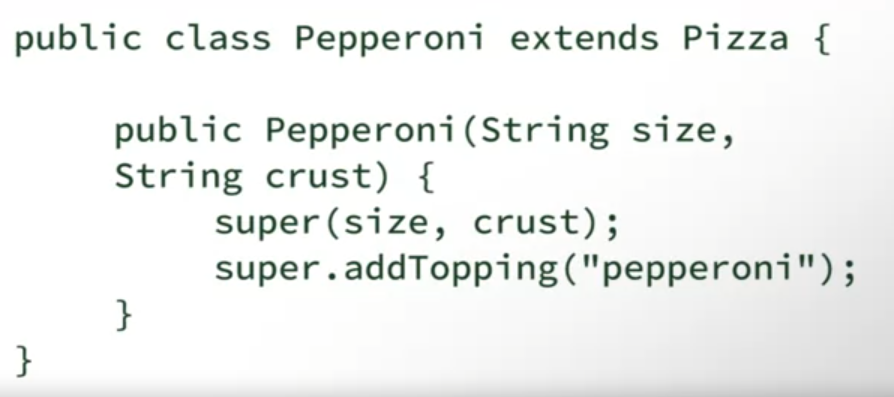
1st indication of improper use of generalization:





Example of misuse of inheritance





**Given the following superclass-subclass pairings, which one is a proper use of inheritance?**



Superclass: Sandwich. Subclass: Ham; Tuna Salad; Chicken Salad



Superclass: Animal. Subclass: Dog, Cat, Whale

Correct

Nice job! There is probably some behaviour that is common to all of these, and some behaviour that is very different between them.



Superclass: Employee Subclass: Manager, Salesperson, Cashier

Correct

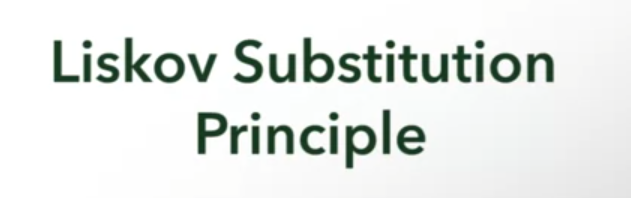
Great! Employee is a generalized type for manager, salesperson, and cashier, but each of these types of employee perform specific functions.

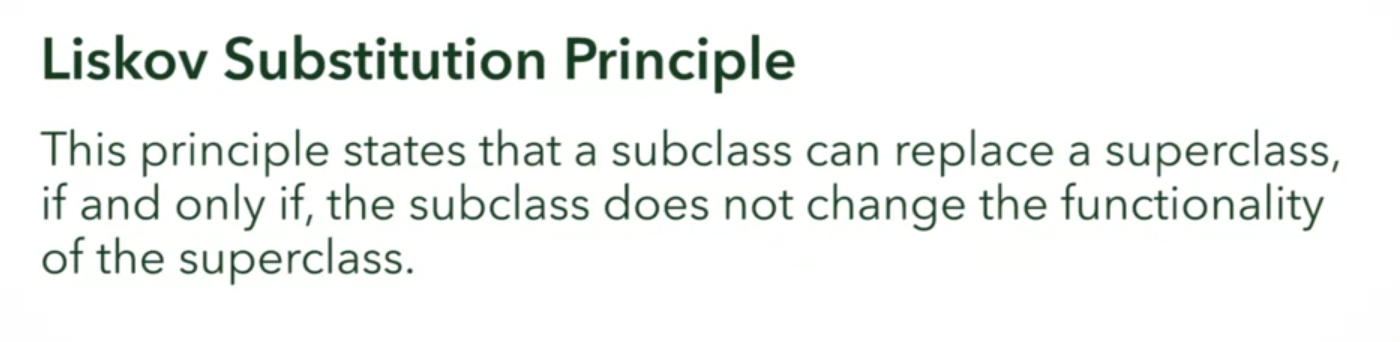


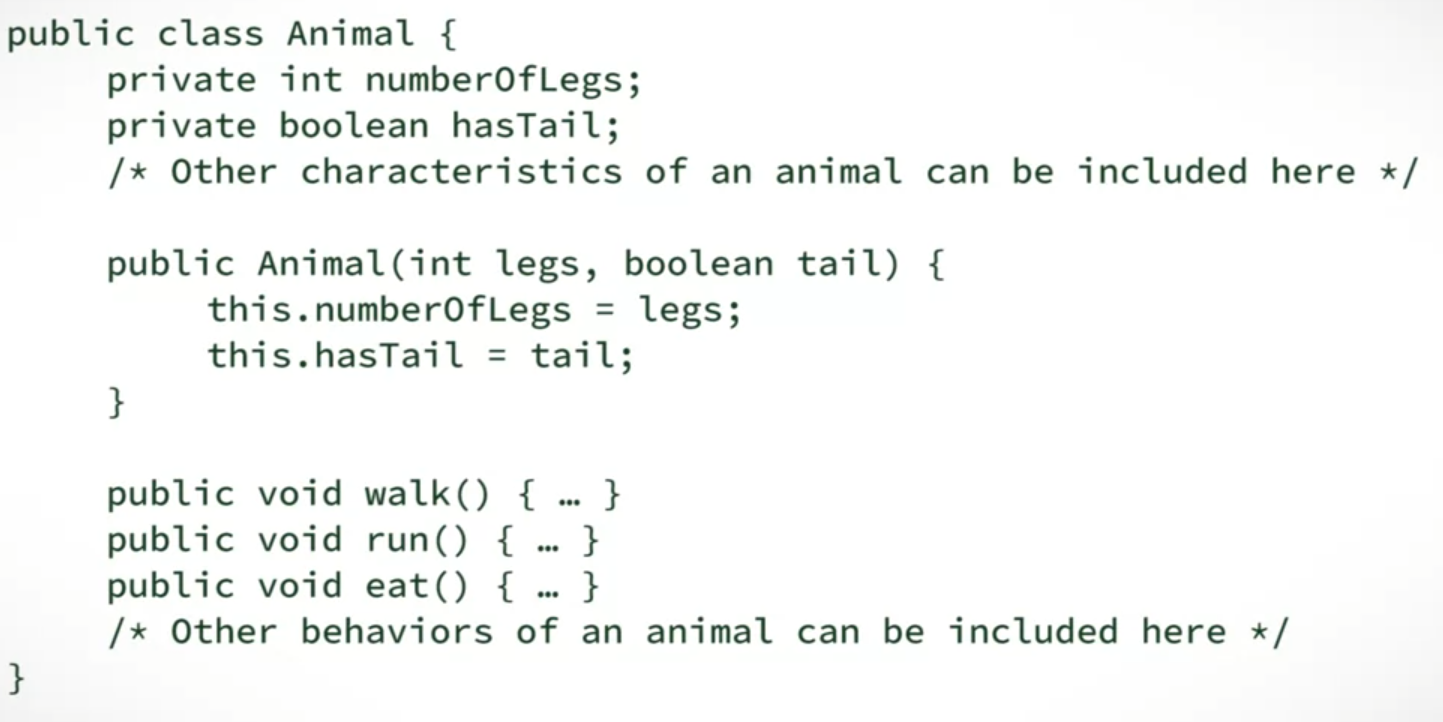
Superclass: Dog Subclass: Husky, Golden Retriever, Poodle

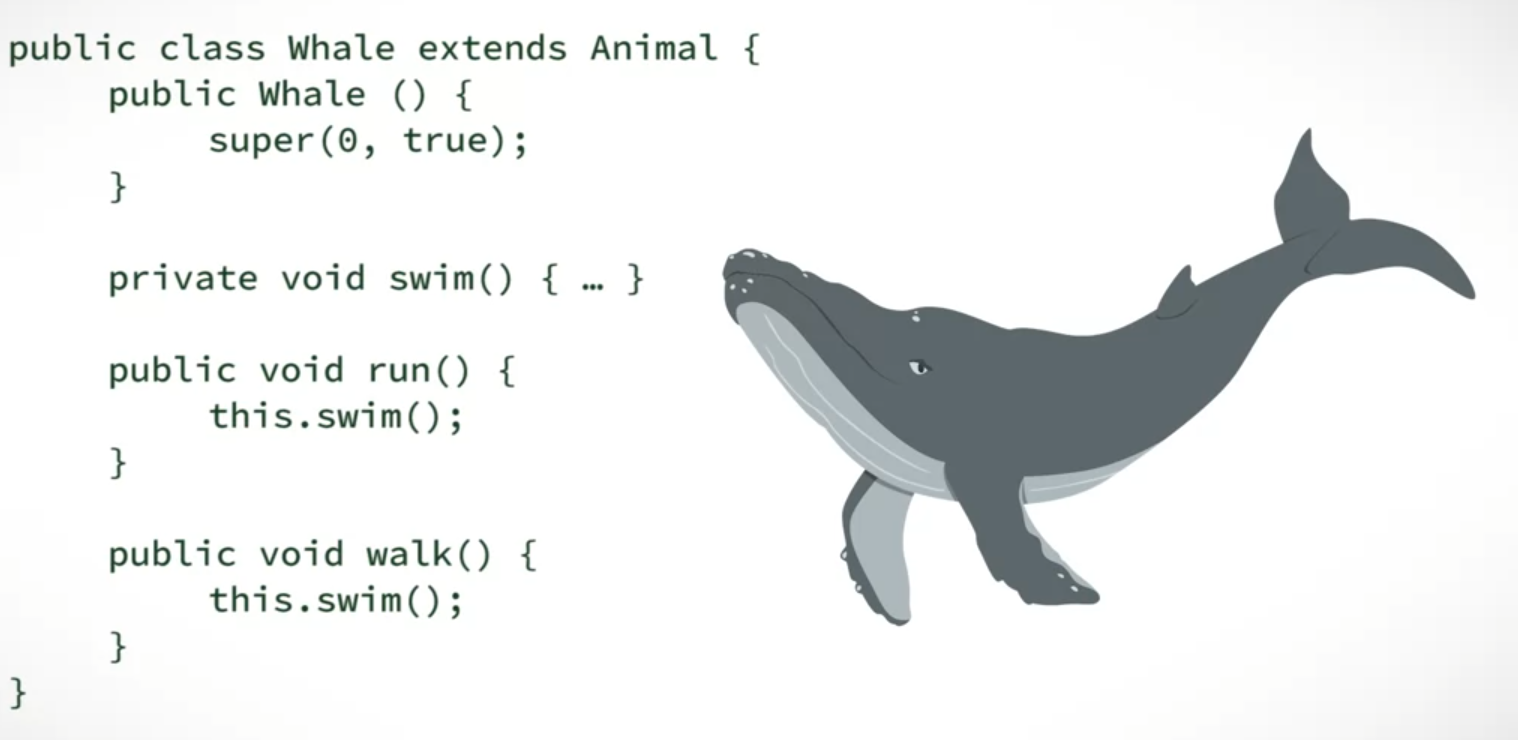
2nd indication of improper use of generalization:

If we violated the:









* Violated the substitution principle since the run() and walk() behavior is replaced!!!

We’ve seen ways in which inheritance can be misused, but in what case is it correct to generalize classes and use inheritance?

**Select the two conditions in which you would be able to properly use the generalization principle for inheritance.**



The subclass has more functionality than the superclass.

Correct

Correct! This is one condition for using a subclass. If there were no specialized behaviour, you could just use the superclass directly.



Subclasses provide and share attributes and behaviors from the same superclass, but each subclass has their own distinct functions.

Correct

Exactly! This is the ideal use of inheritance.



A subclass can replace all of a parent class’ behaviors with totally different responsibilities



Any class should not know too much about the system.

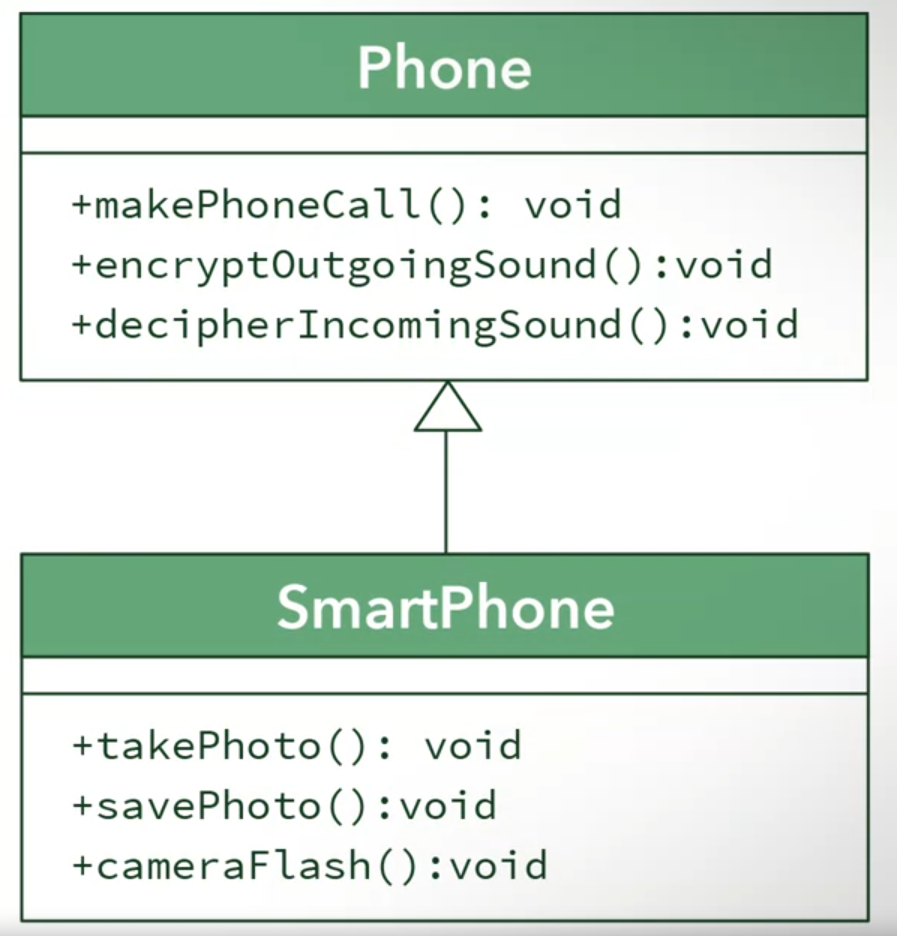
Java Collections Library’s bad inheritance:



* Expected behavior:
  + FILO data structure (First In, Last Out)
  + Peek, pop, push
* But in Java, it inherits from the **vector** class
  + Able to return, retrieve, or insert an element at a specified index
    - NOT EXPECTED FROM A STACK

If inheritance does not suit, consider DECOMPOSITION

Inheritance:



Decomposition:

