LECTURE 27

OBJECT-ORIENTED PROGRAMMING 3

INHERITANCE

MCS 260 Fall 2021

Emily Dumas

REMINDERS

- Read the project 3 description before Wednesday
- Project 3 due 6:00pm central on Fri Nov 5
- Worksheet 10 will be posted this afternoon
- Homework 9 due at 10am tomorrow

GOALS

- Continue working on Rectangle and Circle classes
- Add additional operator overloading
- Add a subclass

MORE OVERLOADING

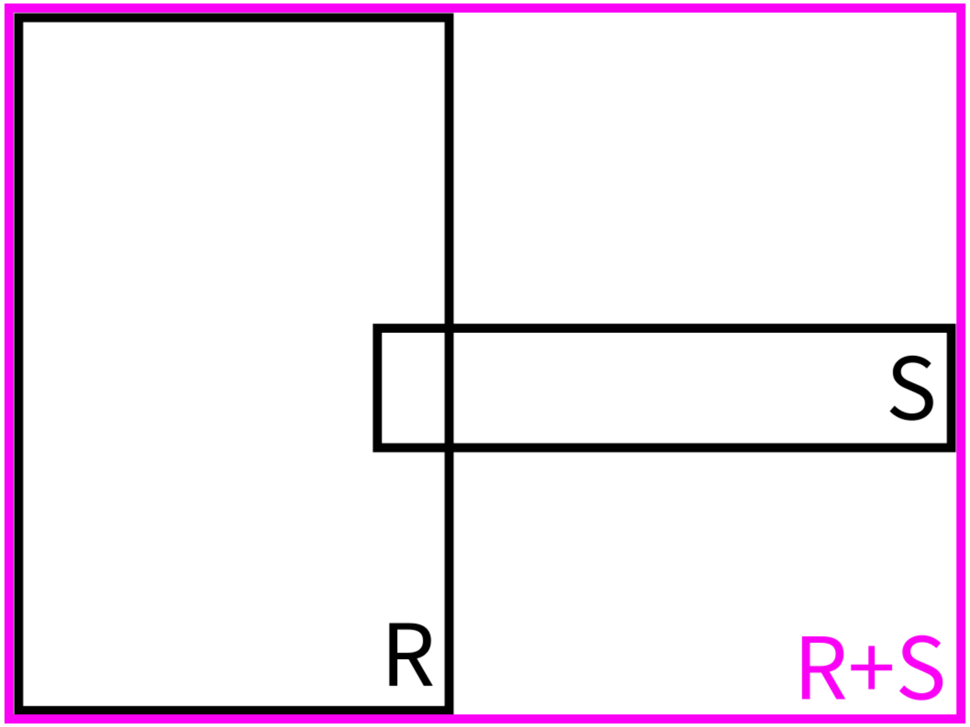
Recall **operator overloading** means writing code to give built-in operators custom behavior when applied to your classes.

Last time: Custom equality test with `__eq__`.

Now: Custom addition with `__add__`.

How should we add two instances of Rectangle?

Idea: Define `R+S` to be the smallest rectangle that contains both `R` and `S`.



INHERITANCE

Complex programs may have many classes.

Often, some classes have a "is-a" relationship: One represents a more specific type of object than another.

e.g. Dresser is a FurnitureItem

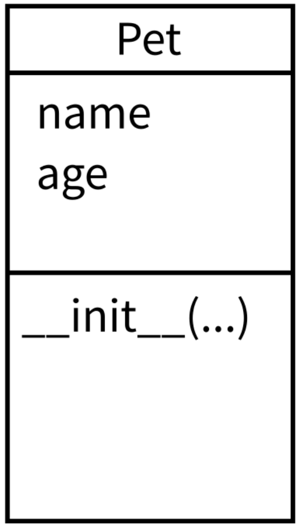
More restrictive classes can have specialized functions (e.g. `open_drawer(idx)`) and attributes (e.g. `ndrawers`).

In OOP, is-a relationships are formalized through **inheritance**. The more specific class is a **subclass** of the more general one.

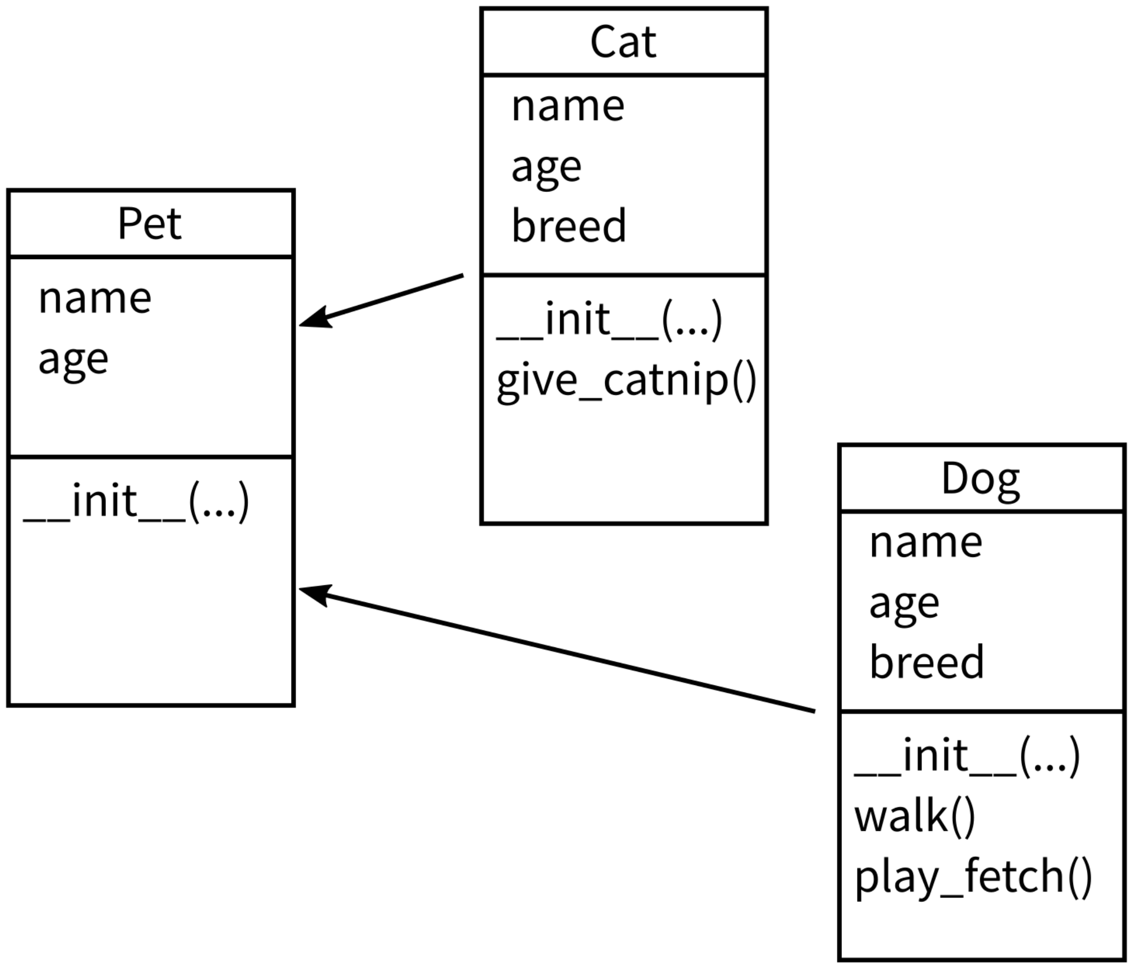
Subclasses inherit all methods and attributes from their superclass, but these can be changed or added to in the subclass definition.

Syntax: `class Dresser(FurnitureItem):`

CLASS HIERARCHY EXAMPLE



CLASS HIERARCHY EXAMPLE



IN GEOM MODULE?

Circle and Rectangle share a lot of behavior—should both be subclasses of another class?

This is worth considering, but we won't do it today.

What if we want to add a class Square? Since any square is a rectangle, we should make Square a subclass of Rectangle.

SUPER()

In a method of a subclass, `super()` returns an modified view of the current object that behaves like an instance of the superclass.

e.g. In a `Square` object, `super()` returns a version of the same object that will act like a `Rectangle`.

`super()` is often used to call the superclass constructor.

__CLASS__

Every object has an attribute `__class__` that refers to its class.

In a method body, `self.__class__.__name__` gives the name of the class as a string.

REFERENCES

- In *Downey*:
    - Chapter 17 discusses classes, objects, and methods
- Object-oriented programming is discussed in general terms in Section 6.5 of Brookshear & Brylow.

REVISION HISTORY

- 2021-10-25 Initial publication