Combining Objects (Inheritance I)

- A class can use another class as an instance variable (sometimes called a *field*).
- This called ***object composition***.
- Use this when you would say "An object of class A ***has an*** object of class B."
  - A dog has an owner.
  - A car has an engine.
  - A student has an advisor.
  - A line segment has a starting point and an ending point.
  - A ComboPolygon has an array of Polygons.

```
public class Dog {
  private String owner;

  // more stuff here
};
```

```
public class Polygon {
  // things here
};

public class ComboPolygon {
  private Polygon[] polys;
};
```

- A class can use another class as an instance variable (sometimes called a *field*).
- This called ***object composition***.
- Use this when you would say "An object of class A ***has an*** object of class B."
  - A dog has an owner.
  - A car has an engine.
  - A student has an advisor.
  - A line segment has a starting point and an ending point.
  - A ComboPolygon has an array of Polygons.
- Another way to express this is that one object is a ***component*** of another object.

- Object composition is also known as a "has-a" relationship.
- A different kind of relationship is an "is-a" relationship.
- Use an "is-a" relationship to express when ***a class is a specific kind of another class.***
  - A poodle is a specific kind of dog.
  - A racecar is a specific kind of car.
  - A textbook is a specific kind of book.
- This concept is called ***inheritance***.

Inheritance (is-a) versus composition (has-a)

- Inheritance expresses that one class can do everything another class can do, plus more:
  - A racecar is just a car that can also drive extra fast around a race track.
- Composition expresses that one class is a component of another class:
  - An engine is a piece of a car.

- When a derived class inherits from a base class:
  - Inside the derived class, the derived class has access to all the public and protected members of the base class.
  - Inside the derived class, the derived class cannot access private members of the base class.
  - Outside the derived class, the derived class has all the same public members as the base class has, plus anything public declared in the derived class.
    - except constructors

- Start with the Parrot class.
  - Add a method for the parrot to sleep, so it can regain its energy.
- Create a PetParrot class that inherits from parrot.
  - A PetParrot should be able to do everything a Parrot can do, plus:
  - It has a name that the user should be able to set.
  - It should have the ability to talk, which decreases its energy (how will you change the PetParrot's energy?)