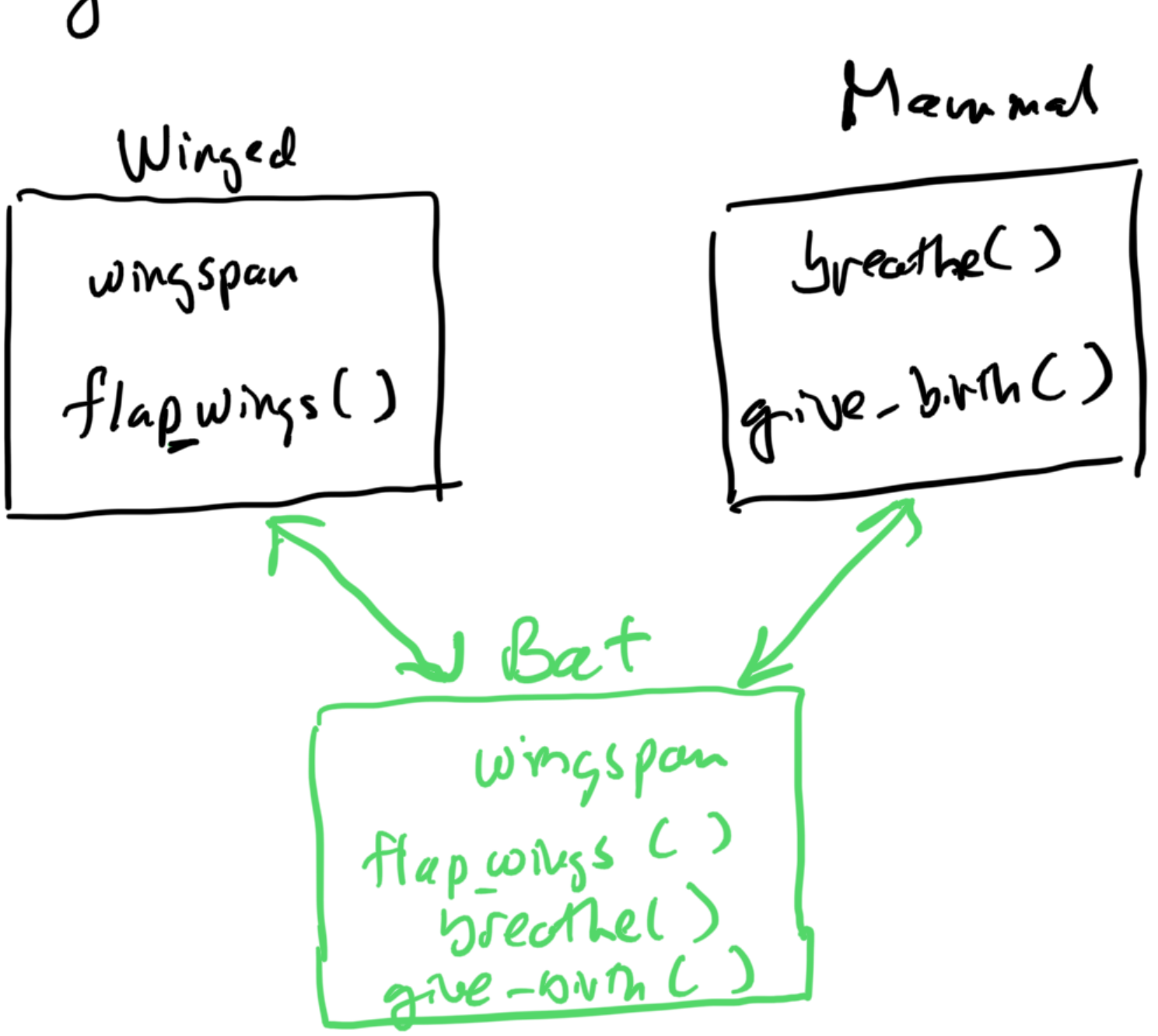
Multiple Inheritance

A class can inherit from more than one base class... this is a "good thing"™. One has to be just a bit careful, though.




```
class Bat(Winged, Mammal):
```

inherits from both base classes.

```
    # What if the base
```

Big Gotcha ..

Classes each contain a common method name (like they each have a method called `blink()`). Which one gets used? or, is this a conflict that causes an error to be thrown?

Let's test this out using the example code in `multiple.py`

Bottom Line: PyTorch will use the method from the first base class.

Mixin Classes

Now that we know that we can inherit from multiple classes, we can do some cool things to make our code more transparent/encapsulated from the point of view of the end user.

Example:

① We inherit from a base class

→ TransportMode

② We then want to create multiple base classes:

→ Truck

→ FlyingCar

For each of those classes, we want to add a new method, called

go()

which is not part of the base class

For Truck, go() → drive()

For FlyingCar, go() → drive() · $\frac{1}{2}$
+ fly() · $\frac{1}{2}$

For the end user, we want them to be able to just write.

```
S = Truck(...)
f = FlyingCar(...)

S.go(100)
f.go(100)
```

and have it all just work!!

To do this, we can define some "mixin" classes that we can inherit as part of Truck() and FlyingCar() as appropriate.

Ex.

```
class DrivingMixin:
    def drive(self, distance)
        ⋮ do stuff
```

```
class FlyingMixin:
    def fly(self, distance, altitude)
        ⋮ do stuff.
```

```
Class Truck(TransportMode,
            DrivingMixin):
    ⋮
    def go(self, distance)
        self.drive(distance)
```

:

class FlyingCar(TransportMode, DrivingMixin, FlyingMixin):

:

def go(self, distance):
    self.fly(distance/2, self.altitude)
    self.drive(distance/2)

See multiple_mixin.py for a fully developed example