# 4. Inheritance :: The Big Picture

The "big picture" view of **inheritance** is that it is employed in OO design when the design of one class (the **subclass**) will be based on the design of another class (the **superclass**). The major advantage is that inheritance **facilitates code re-use** (we will not have to write an entirely new *Square* class when we specify that *Square* is a subclass of *Rectangle*).

Before we being discussing how inheritance is done in Java, let's take another example to make sure the concept is clear. Let's suppose we are designing a program which involves mammals. A mammal is a specific type of living organism and there are many different types of mammals. However, all mammals have specific attributes in common:

**mam·mal**

/'maməl/

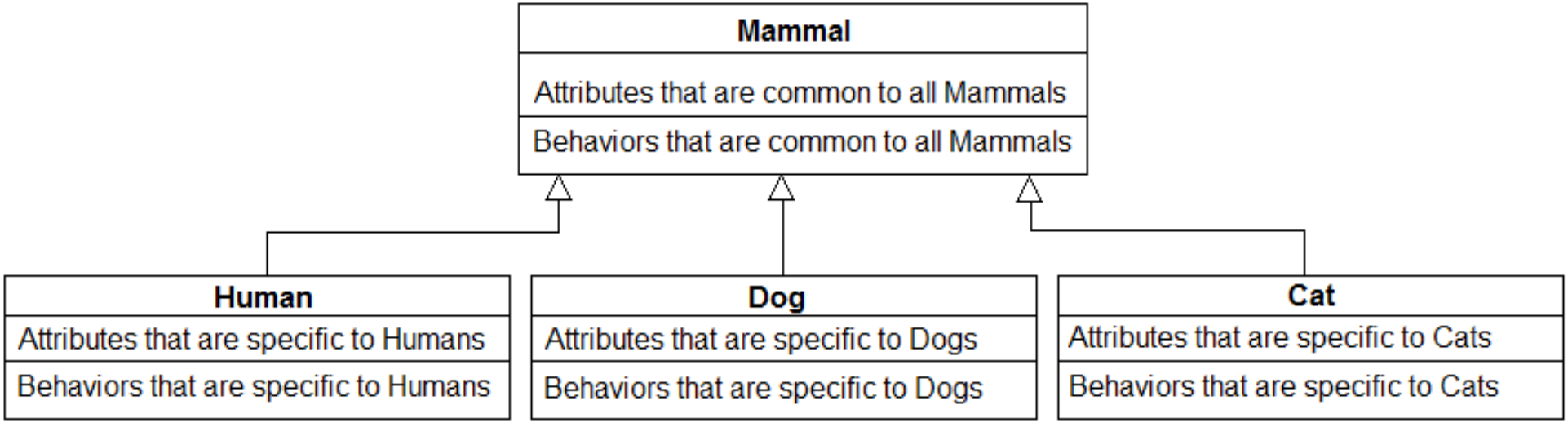
*noun*

1. a warm-blooded vertebrate animal of a class that is distinguished by the possession of hair or fur, the secretion of milk by females for the nourishment of the young, and (typically) the birth of live young.

So the defining characteristics are: (1) warm-blooded; (2) vertebrate; (3) hair or fur; and (4) milk secretion. If you think about humans, cats, and dogs, these characteristics are all present, which is why humans, cats, and dogs are all considered to be mammals. However, humans are neither cats nor dogs, and cats and dogs are distinctly different critters, so obviously there are differences among the three.

## 4. Inheritance :: The Big Picture (continued)

If were going to model these relationships in an OO design, we could design *Mammal*, *Human*, *Dog*, and *Cat* classes. Since the design of the *Human*, *Dog*, and *Cat* classes will be based on the design of the *Mammal* class, we have this inheritance relationship:




The *Mammal* class would declare attributes (data members, instance variables) and behaviors (method members, instance methods) that all *Mammal*s have in common.

Each of the *Human*, *Dog*, and *Cat* classes would declare only those attributes are behaviors that are specific to the category of *Mammal*.

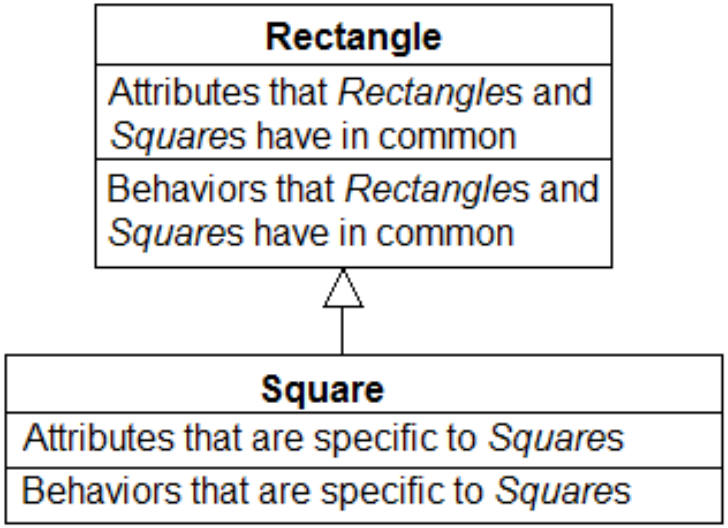
Furthermore, because *Human* is a subclass of *Mammal* (which is a superclass of *Human*) then the *Human* class will inherit certain attributes and behaviors from the *Mammal* class—this is what we mean by **code re-use**. Likewise, the *Dog* and *Cat* classes will also **inherit** those same attributes and behaviors from the *Mammal* class.

## 4. Inheritance :: The Big Picture (continued)

If one of the attributes of the *Mammal* class is a boolean variable *hasHair* which is set to **true** is the organism has hair, then there is no reason to declare *hasHair* in the *Human*, *Dog*, and *Cat* classes. Since they are all *Mammal*s we **know** they have hair and it would only be a duplication of code from the *Mammal* class to redeclare *hasHair* in each of the subclasses.

# 4. Inheritance :: The Big Picture (continued)

In terms of *Rectangle* and *Square* we have this inheritance relationship:




The *Rectangle* class would declare the attributes and behaviors that make a *Rectangle* a *Rectangle* (and since a *Square* **is a** *Rectangle* these are attributes and behaviors that also apply to *Square*s): *mX*, *mY*, *mWidth*, and *mHeight* instance variables; *getX*(), *getY*(), *setX*(), *setY*(), and *move*() method members.

The *Square* class would **inherit** from the *Rectangle* class all of those attributes and behaviors that *Square*s have in common with *Rectangle*s. **Only** those attributes and behaviors that **are specific** to *Square*s would be declared in the *Square* class.