Inheritance

In our daily lives, we talk about and classify the many things around us. We know that the world has things like "dogs" and "cars" and "food" and we are familiar with talking about these things as classes ("dogs are animals that have four legs and people have them as pets and they bark, etc"). We are also used to thinking of things as specific objects ("When I was growing up I had a beagle named Buddy and he liked to chase rabbits.") This is related to how we represent things in the Java programming language, namely we distinguish between classes of things and instances of those classes, namely objects.

In the real world, we are comfortable talking about classes of things at different levels. We talk about dogs and animals and beagles as classes. But notice that there is something subtle going on here, which is that these classes are closely related to each other. Beagles are dogs, and dogs are animals. These relationships are very important and in order to talk sensibly in our daily lives about things in the world we need to be clear on what classes they belong to and how the classes are related to each other. Buddy the beagle was a dog, but he was also an animal. Certain things I might say about Buddy might make more sense in thinking about Buddy as an animal, than thinking about him as a dog or beagle. For example, when I state the fact that Buddy was born in 1966, this statement is more about his being an animal than about him a dog or a beagle. (Being born is something animals do in general, not something specific to dogs or beagles.) Similarly, if I say that Buddy barked, this is related to him being a dog (not an animial or a beagle). If I were to describe a beagle class to someone who doesn't know what a beagle is, I would never think of including breathing or being born as part of the "definition". Dogs *automatically* "inherit" these from the class animal. Beagles in turn automatically inherits such things from the class dog.

How does such class organization work in an object oriented programming language, in particular, Java? Java allows you to derive new classes from existing classes. When we define a class in Java, we specify certain fields and methods. We can then derive a new class from this existing class. We may introduce entirely new fields and methods into the new class, or some of the fields or methods of the new class may be given the same names as those of an existing class (but change the body a method if we wish), or we can let the new class just inherit (unchanged) the fields or methods of the existing class. We will examine these choices over the next few lectures.

Vocabulary

If we have a class `Dog` and we define a new class `Beagle` which *extends* the class `Dog`, we would say that `Dog` is the *base class* or *super class* or *parent class* and `Beagle` is the *subclass* or *derived class* or *extended class*. We say that a subclass *inherits* the fields and methods of the superclass.

Example: a `Progression` class

I spend the rest of this lecture on the example of the `Progression` class which is given in detail in the course textbook (Goodrich & Tommassia, 4th edition. p. 68–75.) You should now read the details in the textbook, as I will not reproduce them here. Some of the details and terminology will not sense yet, but will make sense after the next few lectures.