Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-five)

5.1: Prologue

It is very natural to define things in the real world in terms of other things. For example, I might define a Persian cat as a cat that has a ridiculous amount of fur. I don't need to tell you all about what a cat is in general; this new concept of a Persian cat *inherits* all the properties of a generic cat, and then I add a bit more information. Similarly, a house is a single-family dwelling; a dwelling is a building in which people live. Here I have defined a concept in terms of another concept, which is in turn defined in terms of a third concept. Each definition adds information to the more generic concepts.

Note that a house *is* a building. It's perfectly appropriate to speak of a street full of buildings even if some of them are houses, some apartment houses, and some stores. Whenever you define something as a specialization of something else in this way, it is appropriate to ignore the specialization when linguistic convenience demands.

Java classes have one huge advantage over C structs and Visual Basic TYPEs that we haven't yet discussed. In Java, you can define a new class in terms of a previously defined class. For example, if I have a class named Cat, I can define a class PersianCat that is based on Cat, but has ridiculous amounts of fur. Furthermore, when appropriate, I can refer to a PersianCat object as a Cat, because a PersianCat *is* a cat.

Now we're going to explore this concept. We'll see how this ability to specialize classes, and then to refer to them generically, is really the cornerstone of OOP (object-oriented programming).