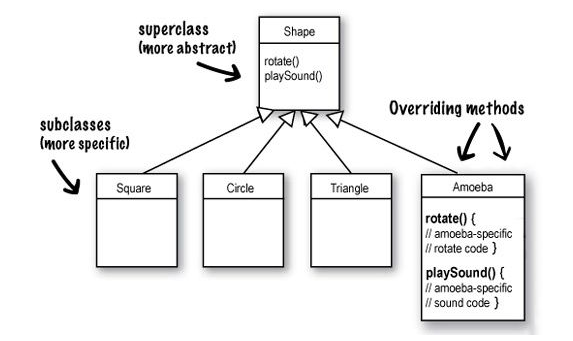
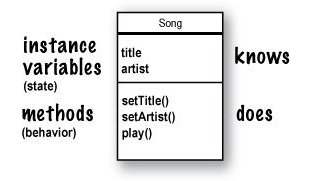
**B”H**

**Chapter 2**

* All Java code is defined in a class.
* Chapter 2 has an awesome example of procedural vs. OO programming example. Memorize this one.



* Some benefits off OO based on above (more to come):
  + Don’t need to touch the tested, working, compiled code - just to add a new feature
  + Helps you design in a more natural way. Things have a way of evolving.
  + The data and the methods that operate on that data are together in one class.
  + Reusing code in other applications. When you write a new class, you can make it flexible enough to be used in something new, later.



* Think of instance as another way of saying object.
* When you design a class, you think about the data an object will need to know about itself, and you also design the methods that operate on that data.
* A class is not an object (but it’s used to construct them). A class is a blueprint for an object.
* The dot operator (.) gives you access to an object’s state and behavior (instance variables and methods).
* The two uses of **main()**:
  + to test your real class
  + to launch/start your Java application
* A real Java application is nothing but objects talking to other objects.
* There isn’t a concept of ‘global’ variables and methods in a Java OO program. In practical use, however, there are times when you want a method (or a constant) to be available to any code running in any part of your program (like the random() method or pi).

You’ll learn in Chapter 10 that marking a method as **public** and **static** makes it behave much like a ‘global’. Any code, in any class of your application, can access a public static method. And if you mark a variable as **public**, **static**, and **final** – you have essentially made a globally-available constant.

Keep in mind that these static (global-like) things are the exception rather than the rule in Java. They represent a very special case, where you don’t have multiple instances/ objects.

* It would be a big pain to deliver a huge bunch of individual files to your end-users, but you won’t have to. You can put all of your application files into a Java Archive – a .jar file – that’s based on the pkzip format. In the jar file, you can include a simple text file formatted as something called a manifest, that defines which class in that jar holds the main() method that should run.

.