public class Person
{
    // instance variables
    private String name;
    private int age;
    public boolean happy;

    // constructor
    public Person(String first_last, int number)
    {
        name = first_last;
        age = number;
        happy = true; // everyone starts out happy
    }

    // toString()

    // happy accessor ("getter")

    // happy mutator (setter)
}
2) Write some code to instantiate an object of the Person class.

3) Suppose you are writing some code as part of a driver class that uses the Person class. You have a variable named jackie referring to an object of the Person class.
   a) Show how to make jackie not happy using the mutator (setter) method.

   b) Show how to make jackie not happy by accessing the variable directly.

   c) Why is it possible to access the variable directly? What correction do you need to make to the code to support encapsulation?

4) Write the definition for another method of the Person class, adNauseum(), that has one parameter, an integer n, and prints out the state of the person n times ("happy" or "sad"). The method should not return anything.
   For example, after the code from the previous question,
   Jackie.adNauseum(4);
   would generate the output:
   sad sad sad sad
5) Sketch a UML class diagram for the Person class. Include all the instance variables and methods from previous questions.

More Practice: TRY THIS AGAIN WITH DOGS!

Create a class for a Dog, similar to the Person class, above. Instead of having a boolean variable to encode happiness, have an integer that encodes the dog’s state (e.g.: 0=sleeping, 1=eating, 2=running, etc – come up with a short list of states and stick to it). Include a constructor, toString(), and accessor and mutator methods for all its variables. Come up with questions similar to 2, above, for the Dog class. For example, write some code that would set a Dog object’s state to sleeping or running, depending on the Dog’s age. Draw the UML class diagram for the Dog class.