**Animal House - Part 1**

You're an animal lover and very fond of keeping track of things. Write a program that will keep track of all the animals you have in your house.

All instance ***variables*** should be private. Add public getter / setter methods when required. It is considered best practice to only release information to the "public" (clients of your class) as necessary, and "encapsulate" the rest.

1. Create a BlueJ project called AnimalHouse. Begin with a class called Animal. Animal will **have** the following:
   1. Instance variables String name and int birthYear . **Write getters and setters for these**.
   2. An Animal may **have** one or many Toys (more info to follow), and **should have a method to add a new Toy** to its collection (think of a datatype that holds many things).
   3. An Animal may also **have** a friend of type Animal. The friend variable should initially start null, and **a method should be created to set the friend** .
   4. A static integer variable representing the currentYear. Initialize it to the current year. Note static variables are NOT initialized in constructors. Constructors only initialize *instance* variables.
   5. A public method int getAge() that returns this Animal's age in years, given the currentYear and the Animal object's birthYear (don't worry about the months).

**/\* static variables should be accessed through the class. You shouldn't access the current year using *this*, because *this* refers to *this object* \*/**

* 1. A public method String toString() that returns a printable String containing this Animal's info, like the following:

Hello, I am <name>. I am <age> years old.

If this Animal has a friend, add the following line:

I have a friend named <friend's name>.

Otherwise, add the line "I have no friends" :(

Finally, add the line "I have the following toys:”

concatenated with this Animal's toys.

All classes should have a well-written toString() method that allows its objects' state to be printed in a useful manner (rather than something like House@677327b6, which is the format inherited from Object – type/@ symbol/hex representation of the object's memory location). Very useful when debugging!

**Note that, for a variable Animal a, the line System.out.println(a) is no different than the line System.out.println(a.toString()), as the compiler adds the toString() call for you if you omit it (when printing an object).**

* 1. Review your class. If you have done everything correctly, according to the instructions above, this class should have 4 PIVs and a 2 parameter constructor, among other things.

1. Write two classes Dog and Cat that extend Animal.
   1. Dog should have a boolean goodBoy instance variable and appropriate constructor.
   2. Cat should have an int numLives instance variable. Cat should have two constructors:
      1. Three parameters: String name, int birthYear, and int numLives. Should contain a super() call.
      2. Two parameters: String name and int birthYear.

The two-parameter constructor should utilize the three-parameter constructor (rather than repeating all the code in each) with a call to this(name, birthYear, 9) – in other words, if a Cat object is created without specifying the number of lives it has, it will be given 9 lives by default.

Without a period, this() refers to *this* object's constructor. This concept is referred to as "constructor chaining" – chaining constructors together, rather than repeating all the code in each. Note that this() and super() are mutually exclusive (more info in the PPTs).

* 1. Override (don’t forget @override) the toString() method, adding the information specific to Dogs and Cats. Use super to re-use what is already done in Animal.