**Animal House - Part 2**

1. Write a class called Toy that has the following:
   1. A String name instance variable and applicable constructor.
   2. An overridden toString() method in the form of "A <name>".
   3. An overridden equals() method (also inherited from Object), that returns true if this Toy has the same name as the parameter. Check the powerpoints for info on overriding the equals() method.

/\* Recall that when comparing Strings, you can't check for equivalency with the comparison (==) operator. Instead, you must use the equals() method. When you do this, you are calling the String class' overriden equals() method! \*/

**Add the @Override annotation** above the method signature as a sanity check – the compiler will warn you if you are not actually overriding a method (examples in the powerpoints).

1. Write a class called House that has the following:
   1. A House instance can have one or many animals.
   2. A printAnimals() method that prints the info on all the Animals in the house. There should only be one print statement inside the **for-each loop**.
   3. Add a ***private*** method to the House class called cleanHouse(). Yes, private! This method will remove any duplicate toys that a *particular Animal* possesses (two different Animals can have a similar toy). When testing this, you should use print statements to print out the list of toys at the beginning of the method and then another print statement to print out the list of toys at the end of the method. This will help you to know if cleanHouse() is working properly!
      1. Use the ArrayList class' indexOf() and lastIndexOf() methods to check for (then remove) duplicate Toys. Google is your friend - Java ArrayList API ! **/\* useful for displaying polymorphism at work \*/**

Use the following loop as a beginning point:

for (Animal a : animals)

{

ArrayList<Toy> toys = a.getToys(); //one Animal's toys each iteration

for (int i = 0; i < toys.size(); i++)

/\* Add your code here \*/

}

If your ArrayList is properly parameterized (e.g. ArrayList<Toy> toys), the indexOf() and lastIndexOf() methods will use a call to Toy's overridden equals() method to determine if a particular object exists in the list. In other words, the indexOf() method returns the index where a call to the overridden equals() method for that object returns true. Gaaaaaah!

If your ArrayList WASN'T parameterized (e.g. ArrayList toys), list elements would be type Object, and Object's equals() method would be called (which simply uses the == operator to check if they are the same object), rather than Toy's overridden method.

* 1. Add a call to cleanHouse() as the first line in the printAnimals() method.

1. A Runner class with a main() method has been provided. Done properly, your output should look like the output below. Fix any mistakes you may have.

Same toy? false

Equal toys? true

Hello, I am Sparky. I am 13 years old.

I have a friend named Fluffy

I have the following toys: [A Busy bee, A Bone, A Ball]

I am a good boy.

Hello, I am Fluffy. I am 10 years old.

I have a friend named Sparky

I have the following toys: [A Mouse]

I have 9 lives.