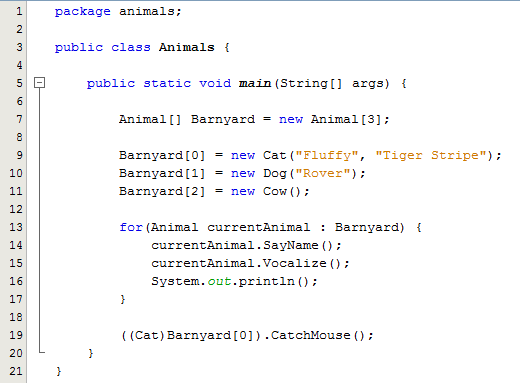
**Java lesson two**

1. Base class Animal with only Vocalize and no abstracting
2. Cat extends Animal
3. Change Meow to Vocalize (Main winds up with error)
4. Use Refactor to change Meow to Vocalize
5. Add @Override directive
6. Copy Dog from HelloWorld
7. Dog extends Animal (do not change Bark to Vocalize)
8. Use abstract to force Dog to implement Vocalize
9. Create Cow class
10. Implement SayName in Animal so that Cow doesn't have to
11. Review variables (pointers) and types
12. Barnyard array
13. for loop
14. Cast Animal to Cat

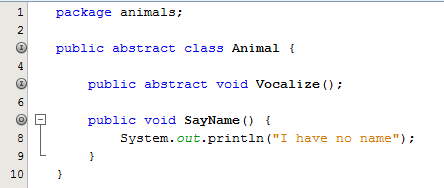
Additional Goals (to be accomplished by next week)

1. Add a method to Cat that returns its CoatColor.
   1. The method MUST NOT print the CoatColor to System.out - it must instead send the CoatColor back to the caller as a **String**. See the **Add** method in Lesson One's Dog class for an example to follow.
2. Add a line to Animals.Main that prints the cat's CoatColor to System.out similar to "The cat has a \_\_\_ coat".
   1. If you want an extra challenge, print the cat's name instead "The cat" (hint: you will have to add a method to the Cat class that returns the name)
3. Add an abstract method to Animal named "DoATrick". Implement DoATrick in each of the sub-classes (which Java will force you to do anyway). You choose what each animal will do when DoATrick is called.
4. Create a **Goat** class that inherits from (extends) Animal.
   1. You choose whether the Goat's name is required (like Cat or Dog) or is not required (like Cow)
   2. Implement Vocalize and DoATrick methods
5. In Animals.Main:
   1. Add a Goat to the Barnyard array (hint: you'll have to increase the size of the array)
   2. Add a call to currentAnimal.DoATrick inside the **for** loop

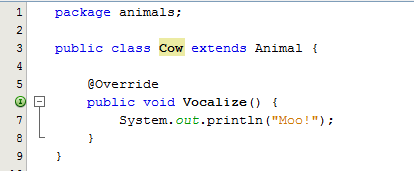
Animals.java



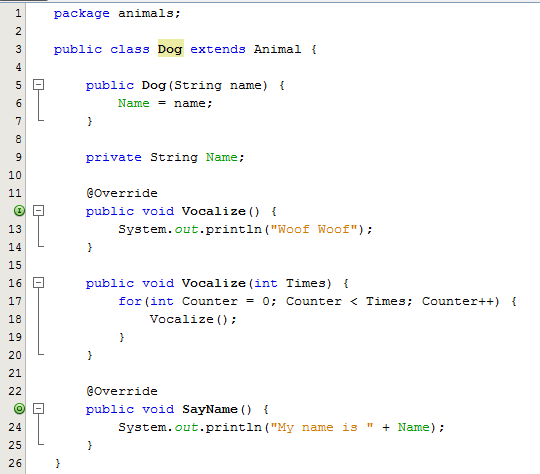
Animal.java



Cow.java



Dog.java



Cat.java

