1 Creating Cats

Given the Animal class, fill in the definition of the Cat class so that it makes a "Meow!" noise when greet() is called. Assume this noise is all caps for kittens (less than 2 years old).

```java
public class Animal {
    protected String name, noise;
    protected int age;
    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = "Huh?";
    }
    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }
    public String greet() {
        return name + ":" + makeNoise();
    }
}

class Cat extends Animal {
}
```

2 Testing Cats

We now want to write tests for our Animal class. Fill in the blanks to test the greet() method.

```java
import static org.junit.Assert.*;
import org.junit.Test;

public class AnimalTest {
    @Test
    public void testGreet() {
        Animal a = new Animal("Pluto", 10);
        Cat c = new Cat("Garfield", 1);
        assertEquals(a.greet(), ________________________);  // (A)
        assertEquals(c.greet(), ________________________);  // (B)
        a = c;
        assertEquals(a.greet(), ________________________);  // (C)
    }
}
```
3 Raining Cats & Dogs

We now have the Dog class! (Assume that the Cat and Dog classes are both in the same file as the Animal class.)

class Dog extends Animal {
    public Dog(String name, int age) {
        super(name, age);
        noise = "Woof!";
    }
    public void playFetch() {
        System.out.println("Fetch, " + name + "!");
    }
}

Consider the following main function in the Animal class. Decide whether each line causes a compile time error, a runtime error, or no error. If a line works correctly, draw a box-and-pointer diagram and/or note what the line prints.

public static void main(String[] args) {
    Cat nyan = new Animal("Nyan Cat", 5); (A)___________________________
    Animal a = new Cat("Olivia Benson", 3); (B)___________________________
    a = new Dog("Fido", 7); (C)___________________________
    System.out.println(a.greet()); (D)___________________________
    a.playFetch(); (E)___________________________
    Dog d1 = a; (F)___________________________
    Dog d2 = (Dog) a; (G)___________________________
    d2.playFetch(); (H)___________________________
    (Dog) a.playFetch(); (I)___________________________
    Animal imposter = new Cat("Pedro", 12); (J)___________________________
    Dog fakeDog = (Dog) imposter; (K)___________________________
    Cat failImposter = new Cat("Jimmy", 21); (L)___________________________
    Dog failDog = (Dog) failImposter; (M)___________________________
}

CS 61B, Fall 2015, Discussion 4: Inheritance
4 Bonus: An Exercise in Inheritance Misery

Cross out any lines that cause compile or runtime errors. What does the main program output after removing those lines?

```java
class A {
    int x = 5;
    public void m1() {System.out.println("Am1-> " + x);}
    public void m2() {System.out.println("Am2-> " + this.x);}
    public void update() {x = 99;}
}

class B extends A {
    int x = 10;
    public void m2() {System.out.println("Bm2-> " + x);}
    public void m3() {System.out.println("Bm3-> " + super.x);}
    public void m4() {System.out.print("Bm4-> "); super.m2();}
}

class C extends B {
    int y = x + 1;
    public void m2() {System.out.println("Cm2-> " + super.x);}
    public void m3() {System.out.println("Cm3-> " + super.super.x);}
    public void m4() {System.out.println("Cm4-> " + y);}
    public void m5() {System.out.println("Cm5-> " + super.y);}
}

class D {
    public static void main (String[] args) {
        A b0 = new B();
        System.out.println(b0.x); (A) ________________
        b0.m1(); (B) ________________
        b0.m2(); (C) ________________
        b0.m3(); (D) ________________
        B b1 = new B();
        b1.m3(); (E) ________________
        b1.m4(); (F) ________________
        A c0 = new C();
        c0.m1(); (G) ________________
        A a1 = (A) c0;
        C c2 = (C) a1;
        c2.m4(); (H) ________________
        ((C) c0).m3(); (I) ________________
        b0.update();
        b0.m1(); (J) ________________
    }
}
```