

---

## 1 Creating Cats

---

Given the following classes, fill in the definition of the `Cat` class so that when `greet()` is called, the label "Cat" (instead of "Animal") is printed to the screen. Assume that a `Cat` will make a "Meow!" noise, and that this is all caps for cats who are less than 5 years old.

```
1 public class Animal {
2     protected String name, noise;
3     protected int age;
4
5     public Animal(String name, int age) {
6         this.name = name;
7         this.age = age;
8         this.noise = "Huh?";
9     }
10
11    public String makeNoise() {
12        if (age < 5) {
13            return noise.toUpperCase();
14        } else {
15            return noise;
16        }
17    }
18
19    public void greet() {
20        System.out.println("Animal " + name + " says: " + makeNoise());
21    }
22 }
```

```
public class Cat extends Animal {
```

```
}
```

## 2 Raining Cats and Dogs

---

Assume that `Animal` and `Cat` are defined as above. What will be printed at each of the indicated lines?

```
1 public class TestAnimals {
2     public static void main(String[] args) {
3         Animal a = new Animal("Pluto", 10);
4         Cat c = new Cat("Garfield", 6);
5         Dog d = new Dog("Fido", 4);
6
7         a.greet();           // (A) _____
8         c.greet();           // (B) _____
9         d.greet();           // (C) _____
10
11        a = c;
12        a.greet();           // (D) _____
13        ((Cat) a).greet();   // (E) _____
14    }
15 }
16
17 public class Dog extends Animal {
18     public Dog(String name, int age) {
19         super(name, age);
20         noise = "Woof!";
21     }
22
23     @Override
24     public void greet() {
25         System.out.println("Dog " + name + " says: " + makeNoise());
26     }
27
28     public void playFetch() {
29         System.out.println("Fetch, " + name + "!");
30     }
31 }
```

Consider what would happen we added the following to the bottom of `main`:

```
a = new Dog("Hieronymus", 10);
d = a;
```

Why would this code produce a compiler error? How could we fix this error?

### 3 An Exercise in Inheritance Misery

Cross out any lines that cause compile-time, and put an X through runtime errors (if any). What does the main program (in class D) output after removing these lines? Note: This problem is deliberately obtuse in order to cover Java corner cases, and is not reflective of how inheritance is typically used. We won't have time in discussion to complete this entire problem.

```
1 class A {
2     public int x = 5;
3     public void m1() {System.out.println("Am1-> " + x);}
4     public void m2() {System.out.println("Am2-> " + this.x);}
5     public void update() {x = 99;}
6 }
7
8 class B extends A {
9     public int x = 10;
10    public void m2() {System.out.println("Bm2-> " + x);}
11    public void m3() {System.out.println("Bm3-> " + super.x);}
12    public void m4() {System.out.print("Bm4-> "); super.m2();}
13 }
14 class C extends B {
15     public int y = x + 1;
16     public void m2() {System.out.println("Cm2-> " + super.x);}
17     public void m3() {System.out.println("Cm3-> " + super.super.x);}
18     public void m4() {System.out.println("Cm4-> " + y);}
19     public void m5() {System.out.println("Cm5-> " + super.y);}
20 }
21 class D {
22     public static void main (String[] args) {
23         B a0 = new A();
24         a0.m1();
25         A b0 = new B();
26         System.out.println(b0.x);
27         b0.m1(); // class B hides a field in class A.
28         b0.m2(); // you should never hide fields.
29         b0.m3(); // as you'll see, it's confusing!
30         B b1 = new B();
31         b1.m3();
32         b1.m4();
33         A c0 = new C();
34         c0.m1();
35         C c1 = (A) new C();
36         A a1 = (A) c0;
37         C c2 = (C) a1;
38         c2.m4();
39         c2.m5();
40         ((C) c0).m3(); // very tricky!
41         (C) c0.m3();
42         b0.update();
43         b0.m1();
44         b0.m2();
45     }
46 }
```