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.

```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 < 5) {
            return noise.toUpperCase();
        } else {
            return noise;
        }
    }

    public void greet() {
        System.out.println("Animal " + name + " says: " + makeNoise());
    }
}

public class Cat extends Animal {
    // Your implementation here
}
```
2 Raining Cats and Dogs

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

```java
public class TestAnimals {
    public static void main(String[] args) {
        Animal a = new Animal("Pluto", 10);
        Cat c = new Cat("Garfield", 6);
        Dog d = new Dog("Fido", 4);

        a.greet(); // (A) ______________________
        c.greet(); // (B) ______________________
        d.greet(); // (C) ______________________

        a = c;
        a.greet(); // (D) ______________________
        ((Cat) a).greet(); // (E) ______________________
    }
}

public class Dog extends Animal {
    public Dog(String name, int age) {
        super(name, age);
        noise = "Woof!";
    }

    @Override
    public void greet() {
        System.out.println("Dog " + name + " says: " + makeNoise());
    }

    public void playFetch() {
        System.out.println("Fetch, " + name + "!");
    }
}
```

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

```java
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.

class A {
    public 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 {
    public 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.println("Bm4-> "); super.m2();}
}

class C extends B {
    public 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) {
        B a0 = new A();
        a0.m1();
        A b0 = new B();
        System.out.println(b0.x);
        b0.m1(); // class B hides a field in class A.
        b0.m2(); // you should never hide fields.
        b0.m3(); // as you’ll see, it’s confusing!
        B b1 = new B();
        b1.m3();
        b1.m4();
        A c0 = new C();
        c0.m1();
        C c1 = (A) new C();
        A a1 = (A) c0;
        C c2 = (C) a1;
        c2.m4();
        c2.m5();
        ((C) c0).m3(); // very tricky!
        (C) c0.m3();
        b0.update();
        b0.m1();
        b0.m2();
    }
}