CS 61B Discussion 4: Inheritance Fall 2015

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).

```
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.

```
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) _____

}
```

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?

```
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) _____
    }
}
```