CS 61B Discussion 4: Inheritance Fall 2014

1 Raining Cats and Dogs

Write out the necessary classes for the code below to compile and produce the given output.

OUTPUT:
Dug wolfs down the food. Woof!
Pluto sniffs the food. Woof!
Salem sniffs the food. Meow.
Nyan Cat wolfs down the food. Meow.

```
class TestAnimals {
    public static void main(String[] args) {
        Dog dug = new Dog("Dug", 5);
        Dog pluto = new Dog("Pluto", 84);
        Cat salem = new Cat("Salem", 18);
        Cat nyancat = new Cat("Nyan Cat", 3);
        Animal[] animals = {dug, pluto, salem, nyancat};
        feed(animals);
    }
    public static void feed(Animal[] animals) {
        for (Animal a : animals) {
            if (a.getAge() < 10) {
                System.out.print(a.getName() + " wolfs down the food. ");
            } else {
                System.out.print(a.getName() + " sniffs the food. ");
            }
            a.makeSound();
        }
    }
}

public abstract class Animal {
        private String _name;
        private int _age;
        public Animal(String name, int age) {
                _name = name;
                _age = age;
        }
        public int getAge() { return _age; }
        public String getName() { return _name; }
        public abstract void makeSound();
}
public class Dog extends Animal {
        public Dog(String name, int age) { super(name, age); }
        //mention (at)Override tag
        public void makeSound() { System.out.println("Woof!"); }
}
public class Cat extends Animal {
        public Cat(String name, int age) { super(name, age); }
        public void makeSound() { System.out.println("Meow."); }
```

```
}
```

Discussion Question: Should `Animal` be an interface or an abstract class?

- Animal should be an abstract class if there are methods that all animals should get and are implemented the same way like getName() and getAge().
- Look how short the Dog and Cat class are above! If this were an interface, we would have to reimplement getAge() and getName().
- A downside to Animal as an abstract class is that Dog and Cat might want to inherit from a different abstract class. You can only inherit from one abstract class, but you can implement multiple interfaces.

Output of Question 2 (Next Page):

```
Am1-> 5
Bm2-> 10
Bm3-> 5
Bm4-> Am2-> 5
Am1-> 5
Cm4-> 11
Bm3-> 5
Am1-> 99
Bm2-> 10
```

Individual outputs are next to their corresponding line and in brackets.

2 Inheritance

Cross out any lines that cause compile-time or runtime errors. What does the main program (in class D) output after removing these lines?

```
1  class A {
2      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      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     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     ^ can't do super.super
19     public void m4() {System.out.println("Cm4-> " + y);}
20     //public void m5() {System.out.println("Cm5-> " + super.y); B class has
           no y
21 }
22 class D {
23     public static void main (String[] args) {
24         //B a0 = new A(); a0 must be B or a subclass of B.
25         //a0.m1(); a0 is invalid
26         A b0 = new B();
27         b0.m1(); [Am1-> 5]
28         b0.m2(); [Bm2-> 10]
29         //b0.m3(); Can only call a method which the static type has.
30         B b1 = new B();
31         b1.m3(); [Bm3-> 5]
32         b1.m4(); [Bm4-> Am2-> 5]
33         A c0 = new C();
34         c0.m1(); [Am1-> 5]
35         //C c1 = (A) new C(); Correct casting syntax. Can't assign c1 to an A.
36         A a1 = (A) c0;
37         C c2 = (C) a1;
38         c2.m4(); [Cm4-> 11]
39         //c2.m5(); m5 is invalid
40         ((C) c0).m3(); Bm3-> 5
41         //(C) c0.m3(); This would cast the result of what the method returns.
42         b0.update();
43         b0.m1(); [Am1-> 99]
44         b0.m2(); [Bm2-> 10]
45     }
46 }
```