CS 61B Spring 2018

Inheritance

Discussion 4: February 6, 2018

1 Creating Cats

1.1 Given the `Animal` class, 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 if the cat is 5 years or older and "MEOW!" if the cat is 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 }
```

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

```
public class Cat extends Animal{
    1 usage
    public Cat(String name, int age) {
        super(name, age);
        this.noise = "Meow!";
    }
    4 usages
    @Override
    public void greet() {
        System.out.println("Cat " + name + " says: " + makeNoise());
    }
}
```

2 Raining Cats and Dogs

2.1 Assume that `Animal` and `Cat` are defined as above. What would Java print on 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) Animal Pluto says: Huh?
8          c.greet();          // (B) Cat Garfield says: Moew!
9          d.greet();          // (C) Dog Fido says: WOOF!
10         a = c;
11         ((Cat) a).greet();  // (D) Cat Garfield says: Moew!
12         a.greet();          // (E) Cat Garfield says: Meow!
13     }
14 }
15
16 public class Dog extends Animal {
17     public Dog(String name, int age) {
18         super(name, age);
19         noise = "Woof!";
20     }
21
22     @Override
23     public void greet() {
24         System.out.println("Dog " + name + " says: " + makeNoise());
25     }
26
27     public void playFetch() {
28         System.out.println("Fetch, " + name + "!");
29     }
30 }
```

Consider what would happen if we added the following to the bottom of `main` under line 12:

```
1  a = new Dog("Spot", 10);
2  d = a;
```

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

```
a = new Dog( name: "Spot",  age: 10);
d = (Dog) a;
```

Because during compilation time, the compiler only checks static type. The static type of a is still Animal, so it cannot be put into the box of a Dog(d)

3 An Exercise in Inheritance Misery *Extra*

3.1 Cross out any lines that cause compile-time errors or cascading errors (failures that occur because of an error that happened earlier in the program), and put an X through runtime errors (if any). Don't just limit your search to main, there could be errors in classes A,B,C. What does `D.main` output after removing these lines?

```
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  class B extends A {
8      public void m2() {System.out.println("Bm2-> " + x);}
9      public void m2(int y) {System.out.println("Bm2y-> " + y);}
10     public void m3() {System.out.println("Bm3-> " + "called");}
11 }
12 class C extends B {
13     public int y = x + 1;
14     public void m2() {System.out.println("Cm2-> " + super.x);}
15     public void m4() {System.out.println("Cm4-> " + super.super.x);}
16     public void m5() {System.out.println("Cm5-> " + y);}
17 }
18 class D {
19     public static void main (String[] args) {
20         B a0 = new A();
21         a0.m1();
22         a0.m2(16);
23         A b0 = new B();
24         System.out.println(b0.x);
25         b0.m1();
26         b0.m2();
27         b0.m2(61);
28         B b1 = new B();
29         b1.m2(61);
30         b1.m3();
31         A c0 = new C();
32         c0.m2();
33         C c1 = (A) new C();
34         A a1 = (A) c0;
35         C c2 = (C) a1;
36         c2.m3();
37         c2.m4();
38         c2.m5();
39         ((C) c0).m3();
40         (C) c0.m3();
41         b0.update();
42         b0.m1();
43     }
44 }
```

overloading

super.x is valid
super.super.x is not

A cannot be put into a smaller box

5

Am1-> 5

Bm2-> 5

Bm2y-> 61

dynamic dispatch is not applicable for overloading

Bm2y-> 61

Bm3-> called

Cm2-> 5

Bm3-> called

Cm4-> 5

Cm5-> 6

Bm3-> called

cannot case the result, complication error

Am1-> 99