CS 61B Spring 2021 — Inheritance

Discussion 4: February 8, 2021

1 Raining Cats and Dogs

(a) Below, four classes are defined. What would Java do after executing the main method in the testAnimals class? Next to each blank, if something is printed write it down. If there is an error, write whether it is a runtime error or compile time error, and then proceed through the rest of the code as if the erroneous line were not there.

```java
public class Animal {
    public String name, noise;
    public int age;

    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = "Huh?";
    }

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

    public void play() {System.out.println("Woo it is so much fun being an animal!")}
}

public class Cat extends Animal {
    public Cat(String name, int age) {
        super(name, age);
        this.noise = "Meow!";
    }

    @Override
    public void greet() {System.out.println("Cat " + name + " says: " + this.noise);}

    public void play(String expr) {System.out.println("Woo it is so much fun being a cat!" + expr)}
}

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: " + this.noise);}
36
37      public void play(int happiness) {
38          if (happiness > 10) {
39              System.out.println("Woo it is so much fun being a dog!")
40          }
41      }
42  }
43
44  public class TestAnimals {
45      public static void main(String[] args) {
46          Animal a = new Animal("Pluto", 10);
47          Cat c = new Cat("Garfield", 6);
48          Dog d = new Dog("Fido", 4);
49          a.greet();              // "Animal Pluto says huh?"
50          c.greet();              // "Cat Garfield says Meow!"
51          d.greet();              // "Dog Fido says Woof!"
52          c.play();               // "Woo it is so much fun being an animal!"
53          c.play(":)")            // "Woo it is so much fun being a cat! :)"
54          a = c;
55          ((Cat) a).greet();      // "Cat Garfield says Meow!"
56          ((Cat) a).play(":D");   // "Woo it is so much fun being a cat! :D"
57          a.play(14);             // Compile time error.
58          ((Dog) a).play(12);     // Runtime error.
59          a.greet();              // "Cat Garfield says Meow!"
60          c = a;                  // Compile time error.
61
62      }
63  }
```

(b) Spoiler alert! There is an error on the last line, line 60. How could we fix this error?

The compilation error on line 60 is because we are trying set `c`, which is of static type `Cat` to be equal to `a`, when the static type of `a` is `Animal`. Even though at runtime, `a` really does have dynamic type `Cat`, the compiler only sees static types so it doesn't believe that this assignment is valid. The compiler only sees that we are trying to set a `Cat` variable to point to an `Animal`, and an `Animal` isn't a `Cat`! One way to fix this error would be to cast a to be a Cat, such that the line reads `c = (Cat) a;`. This would be a valid cast, as the compiler agrees that a variable of static type `Animal` could potentially hold a `Cat`, and so our request is feasible. Because the cast works, then the assignment is also now valid because a variable of static type `Cat` can be told to point to the same thing as another variable of (temporary) static type `Cat`. At runtime, this line will be fine because we were telling the truth: `a` really is a `Cat` dynamically!

2 An Exercise in Inheritance Misery

Cross out any lines that result in compiler errors, as well as subsequent lines that would fail because of the compiler error. 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?

```
public 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;                                       }
}
public class B extends A {
    public void m2() {      System.out.println("Bm2-> " + x);             }
    public void m2(int y) { System.out.println("Bm2y-> " + y);            }
    public void m3() {      System.out.println("Bm3-> " + "called");      }
}
public class C extends B {
    public int y = x + 1;
    public void m2() {      System.out.println("Cm2-> " + super.x);       }
    \\ public void m4() {      System.out.println("Cm4-> " + super.super.x); } can't do super.super
    public void m5() {      System.out.println("Cm5-> " + y);             }
}
public class D {
    public static void main (String[] args) {
        \\ B a0 = new A(); Dynamic type must be B or subclass of B
        \\ a0.m1(); cascading: prev line failed, so a0 can't be initialized
        \\ a0.m2(16); cascading: prev line failed, so a0 can't be initialized
        A b0 = new B();
        System.out.println(b0.x); [prints "5"]
        b0.m1(); [prints "Am1-> 5"]
        b0.m2(); [prints "Bm2-> 5"]
        \\ b0.m2(61); m2 (int y) not defined in static type of b0
        B b1 = new B();
        b1.m2(61); [prints "Bm2y-> 61"]
        b1.m3(); [prints "Bm3-> called"]
        A c0 = new C();
        c0.m2(); [prints "cm2-> 5"]
        \\ C c1 = (A) new C(); Can't assign c1 to an A
        A a1 = (A) c0;
        C c2 = (C) a1;
        c2.m3(); [print Bm3-> called]
        \\ c2.m4(); C.m4() is invalid
        c2.m5(); [print Cm5-> 6]
        ((C) c0).m3(); [print Bm3-> called]
        \\ (C) c0.m2(); NOT RUNTIME ERROR This would cast the result of what the method returns and
    it returns void therefore compile-time error
```

```
        b0.update();
        b0.m1(); [print Am1-> 99]
    }
}
```