CS 61B Spring 2021

Inheritance

Discussion 4: Feburary 8, 2021

1 Raining Cats and Dogs

(a) Below, four classes are defined. What would Java do after executing the main method in the TestAnimal 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. Lines in blue cover casting.

```
1  public class Animal {
2      public String name, noise;
3      public 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 void greet() {System.out.println("Animal " + name + " says: " + this.noise);}
12
13     public void play() {System.out.println("Woo it is so much fun being an animal!")}
14 }
15
16 public class Cat extends Animal {
17     public Cat(String name, int age) {
18         super(name, age);
19         this.noise = "Meow!";
20     }
21
22     @Override
23     public void greet() {System.out.println("Cat " + name + " says: " + this.noise);}
24
25     public void play(String expr) {System.out.println("Woo it is so much fun being a cat!" + expr)}
26 }
27
28 public class Dog extends Animal {
29     public Dog(String name, int age) {
30         super(name, age);
31         noise = "Woof!";
32     }
33
34     @Override
35     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 TestAnimal {
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();          // _____________________
50          c.greet();          // _____________________
51          d.greet();          // _____________________
52          c.play();           // _____________________
53          c.play(":)")       // _____________________
54          a = c;
55          ((Cat) a).greet(); // _____________________
56          ((Cat) a).play(":D"); // _____________________
57          a.play(14);         // _____________________
58          ((Dog) a).play(12); // _____________________
59          a.greet();          // _____________________
60          c = a;             // _____________________
61      }
62  }
```

Declaration Instantiation

Static Dynamic

a. Animal. Cat

c. Cat. Cat

d. Dog. Dog

Compiler only checks static type, runtime checks dynamic

Animal pluto says Huh?

Cat garfield says Meow!

Dog Fido says Woof!

Woo it is so much fun being an animal!

Woo it is so much fun being a cat! :)

Valid as cat is an animal, changes a dynamic type to cat

Cat Garfield says Meow!

Woo it is so much fun being a cat! :D

COMPILER ERROR 1. Animal does not have a static method .play(int x)

RUNTIME ERROR 1, Cat cannot be casted to dog (rt error 1 in notes)

Cat Garfield says Meow!

Compile time error. Cat is-not-a animal

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

Compiler error -> not valid variable assignment.
This can be fixed by casting: c = (cat) a

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?

```
1  public 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  public 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 public 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 public class D {
19     public static void main (String[] args) {
20         B a0 = new A();  Compiler Error - Invalid Assignment - A is not a B
21         a0.m1();
22         a0.m2(16);  Cascading Errors
23         A b0 = new B();  Valid - B is an A
24         System.out.println(b0.x);  5
25         b0.m1();  Am1-> 5.  m1 is a valid method of static type A and m1 inherited from A
26         b0.m2();  Bm2-> 5.  m2 is a valid method of static type A and B.m2() override at runtime
27         b0.m2(61);  Compiler Error - Invalid method call of static type A
28         B b1 = new B();
29         b1.m2(61);  Bm2y -> 61
30         b1.m3();  Bm3 -> called
31         A c0 = new C();  Valid - C is an A
32         c0.m2();  Cm2 -> 5.  m2 is a valid method of static type A and C.m2() override at runtime
33         C c1 = (A) new C();  Compiler Error - Invalid Assignment - A is not a C
34         A a1 = (A) c0;
35         C c2 = (C) a1;
36         c2.m3();  Bm3 -> called
37         c2.m4();  Compiler error -> no valid m4 method
38         c2.m5();  cm5-> 6
39         ((C) c0).m3();  Bm3-> called
40         (C) c0.m2();  Runtime error -> m2() returns void, you can't cast void to (C)
41         b0.update();
```

	Static	Dynamic
a0	B.	A
b0	A	B
b1	B	B
c0	A	C
a1	A	A (c0)
c2	C	C (a1)

Cm2-> 5

```
42          b0.m1();   Am1 -> 99
43      }
44  }
```