CS 61B Spring 2019

Inheritance

Discussion 4: February 11, 2019

JUnit Tests

1.1 Think about the lab you did last week where we did JUnit testing. The following code is a few of these JUnit tests from the lab.

```
1  public class IntListTest {
2
3      @Test
4      public void testList() {
5          IntList one = new IntList(1, null);
6          IntList twoOne = new IntList(2, one);
7          IntList threeTwoOne = new IntList(3, twoOne);
8
9          IntList x = IntList.list(3, 2, 1);
10         assertEquals(threeTwoOne, x);
11     }
12
13     @Test
14     public void testdSquareList() {
15         IntList L = IntList.list(1, 2, 3);
16         IntList.dSquareList(L);
17         assertEquals(IntList.list(1, 4, 9), L);
18     }
19 }
```

What are the advantages and disadvantages of writing JUnit tests?

Creating Cats

2.1 Given the `Animal` class, fill in the definition of the `Cat` class so that when `greet()` is called, "Cat says: Meow!" is printed (instead of "Animal says: Huh?"). Cats less than the ages of 5 can say "MEOW!" instead of "Meow!"

```
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 < 5) {
            return noise.toUpperCase();
        } else {
            return noise;
        }
    }

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

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

Raining Cats and Dogs

3.1 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          a.greet();           // (A)
7          c.greet();           // (B)
8          d.greet();           // (C)
9          a = c;
10         ((Cat) a).greet();   // (D)
11         a.greet();           // (E)
12     }
13 }
14
15 public class Dog extends Animal {
16     public Dog(String name, int age) {
17         super(name, age);
18         noise = "Woof!";
19     }
20
21     @Override
22     public void greet() {
23         System.out.println("Dog " + name + " says: " + makeNoise());
24     }
25
26     public void playFetch() {
27         System.out.println("Fetch, " + name + "!");
28     }
29 }
```

Animal Pluto says: Hug?
Cat Garfield says: Meow!
Dog Fido says: WOOF!
Cat Garfield says: Meow!
Cat Garfield says: Meow!

Consider what would happen if we added the following to the bottom of `main` under line 13:

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

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

Compiler error because the static type of d is Dog and the static type of a is Animal.

Fix it:

d = (Dog) a

把 Animal casting 成 Dog

高级 → 低级

An Exercise in Inheritance Misery *Extra*

4.1 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 }
```

NOT allowed:

Subclass a = new Superclass()

5

Am1-> 5

Bm2-> 5

Bm2y-> 61

Bm3-> Called

Cm2-> 5

Not allowed:

C c1 = (Static Type A)

只能将高层的 Casting 成低层的，反过来不行.

Bm3-> Called

Cm5-> 6

Bm3-> Called

Am1-> 99

	Static Type	Dynamic Type
c_0	A	C
a_1	A	C
c_2	C	C