Static & Dynamic Types

All variables in Java have a static type. This is the type we declared the variable as. Can only declare a variable once → static type doesn't change.

~~X = 5~~ int x = 5;

* Animal a;

OR

* Animal a = ... ;

When we assign a variable (by using =), this affects its dynamic type. The dynamic type is what kind of object the variable actually points to. Can reassign many times → dynamic type can change!

Animal
Dog Cat

* a → Animal box = new Dog(); /* dynamic type of a is Dog */
* a = new Cat(); /* dynamic - cat */

Think of the compiler as a very cautious "proofreader" It keeps an eye out for potential mistakes.

* static type = compile-time type
* dynamic type = run-time type

The compiler must "proofread" with limited info: only knows static type!

It errs on the cautious side... it sometimes thinks there is a mistake even if at runtime there would be no error.

	Static	Dynamic
a	Animal	Dog

* `Animal a = new Dog();`
* `a.bark();` → COMPILE ERROR

(a → Animal)

In cases like this, we have to promise the compiler that at run-time, a will be a dog ⇒ CASTING

```
((Dog) a).bark();
```

If we "break the promise", then we will get an error at run-time instead.

```
*Animal a = new Animal();
((Dog) a).bark();
```

a: S Animal, D Animal

(Dog) a

→ ClassCastException

Important Note:

Casting is simply a promise to the compiler! Casting does not change the type (neither static nor dynamic) of an object!

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 }
```

	Static	Dynamic	x	y
b0	A	B	99	—
b1	B	B	5	—
c0	A	C	5	6
a1	A	C	5	6
c2	C	C	5	6