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.

\* 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 box

\* `a` = 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.

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

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

Important Note:

Casting is simply a promise to the compiler! casting does not change the type (neither static nor dynamic) of an object!
An Exercise in Inheritance Misery *Extra*

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?

```java
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;}//
}
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 int x = 5 //
}
class C extends B {
    public int y = x + 1; // y = 6
    public void m2() {System.out.println("Cm2-> " + super.x);} //
    public void m4() {System.out.println("Cm4-> " + super.super.x);} //
    public void m5() {System.out.println("Cm5-> " + y);} //
}
class D {
    public static void main (String[] args) {
        B a0 = new A(); // compile error
        a0.m1();
        a0.m2(16);
        A a0 = new B();
        System.out.println(a0.x); 5
        b0.m1(); 5
        b0.m2(); // Compile error
        b0.m2(61);
        B b1 = new B();
        b1.m2(61); 61
        b1.m3(); 5
        Bm3-> called
        A c0 = new C();
        c0.m2(); // compile error
        (A) c1 = new C();
        A a1 = (A) c0; //
        C c2 = (C) a1; //
        c2.m3(); 5
        Bm3-> called
        (C) c0.m3();
        c2.m4(); // cascading
        c2.m5(); 6
        Cm5-> called
        ((C) c0).m3(); // Compile error
        b0.update();
        b0.m1(); 99
        Am1-> 99
    }
}
```

Static: x, y  
Dynamic: b0 A B q9 =   
          b1 B B 5 —
          c0 A C 5 6
          al A C 5 6  
          c2 C C 5 6

```
```

Animal

Dog

Poodle

C c0 —

x = 5

y = 6

a1 C C 5 6

c2 C C 5 6
```