Inheritance

3 An Exercise in Inheritance Misery

Cross out any lines that cause compile-time, and put an X through runtime errors (if any). What does the main program (in class D) output after removing these lines? Note: This problem is deliberately obtuse in order to cover Java corner cases, and is not reflective of how inheritance is typically used. We won't have time in discussion to complete this entire problem.

```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 int x = 10;
    public void m2() {System.out.println("Bm2-> " + x);}
    public void m3() {System.out.println("Bm3-> " + super.x);}
    public void m4() {System.out.print("Bm4-> "); super.m2();}
}
class C extends B {
    public int y = x + 1;
    public void m2() {System.out.println("Cm2-> " + super.x);}
    public void m3() {System.out.println("Cm3-> " + super.super.x);}
    public void m4() {System.out.println("Cm4-> " + y);}
    public void m5() {System.out.println("Cm5-> " + super.y);}
}
class D {
    public static void main (String[] args) {
        B a0 = new A();
        a0.m1();
        A b0 = new B();
        System.out.println(b0.x);
        b0.m1();   // class B hides a field in class A.
        b0.m2();   // you should never hide fields.
        b0.m3();   // as you'll see, it's confusing!
        B b1 = new B();
        b1.m3();
        b1.m4();
        A c0 = new C();
        c0.m1();
        C c1 = (A) new C();
        A a1 = (A) c0;
        C c2 = (C) a1;
        c2.m4();
        c2.m5();
        ((C) c0).m3();  // very tricky!
        (C) c0.m3();
        b0.update();
        b0.m1();
        b0.m2();
    }
}
```

```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 int x = 10;
    public void m2() {System.out.println("Bm2-> " + x);}
    public void m3() {System.out.println("Bm3-> " + super.x);}
    public void m4() {System.out.print("Bm4-> "); super.m2();}
}
class C extends B {
    public int y = x + 1;
    public void m2() {System.out.println("Cm2-> " + super.x);}
    //public void m3() {System.out.println("Cm3-> " + super.super.x);} can't
       do super.super
    public void m4() {System.out.println("Cm4-> " + y);}
    //public void m5() {System.out.println("Cm5-> " + super.y);}B class has
    no y
}
class D {
    public static void main (String[] args) {
        // B a0 = new A(); a0 must be B or a subclass of B.
        // a0.m1(); a0 is invalid
        A b0 = new B();
        System.out.println(b0.x); [5] (hidden field: uses static type)
        b0.m1();   [Am1-> 5]
        b0.m2();   [Bm2-> 10]
        // b0.m3();   Can only call a method which the static type has.
        B b1 = new B();
        b1.m3();    [Bm3-> 5]
        b1.m4();    [Bm4-> Am2 -> 5]
        A c0 = new C();
        c0.m1();    [Am1->5]
        // C c1 = (A) new C(); Correct casting syntax. Can't assign c1 to an
           A.
        A a1 = (A) c0;
        C c2 = (C) a1;
        c2.m4(); [Cm4-> 11]
        // c2.m5(); m5 is invalid
        ((C) c0).m3();  Bm3-> 5
        //(C) c0.m3();  This would cast the result of what the method returns.
        b0.update();
        b0.m1();    [Am1-> 99]
        b0.m2();    [Bm2-> 10]
    }
}
```