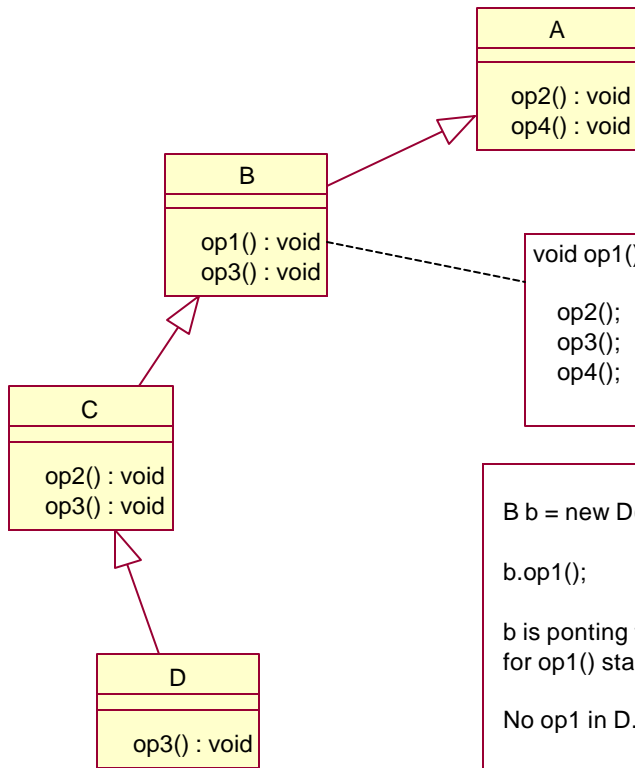


The yo yo problem in deep class hierarchies

Deep hierarchies with overridden methods are very difficult to understand and debug (i.e. difficult to maintain)



```
void op1() {
    op2();
    op3();
    op4();
}
```

```
B b = new D();
```

```
b.op1();
```

b is pointing to an instance of D, therefore the search for op1() starts from D.

No op1 in D. Go to C.

No op1 in C. Go to B. op1 found. op1 of B is invoked.

In op1 of B, op2 is called.

Search for op2 starts from D again! (since b is pointing to an instance of D)

op2 not found in D. Search in C. op3 found in C. op3 of C is executed.

Control returns to op1 of B. Next instruction is to invoke op3.

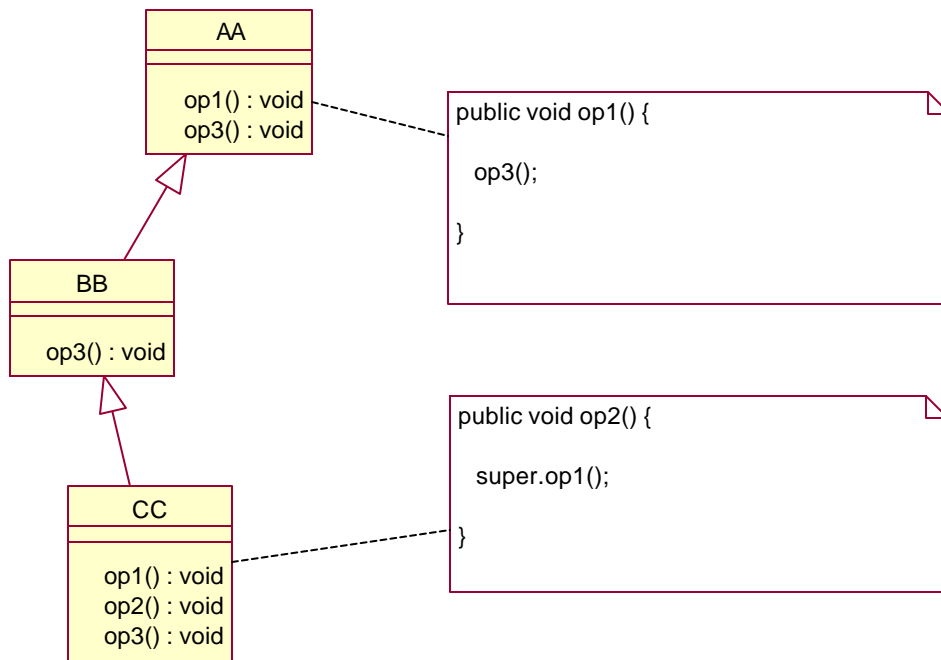
Search for op3 starts from D again!

op3 found in D, therefore op3 in D is executed.

Control returns to op1 of B. Next instruction is to invoke op4. Search starts from D again. op4 is not found in D, C, or B. op4 is finally found in A, and op4 of A is executed.

Simply to figure out which operations are invoked in b.op2(), we have to go up and down the whole hierarchy a number of times, and examine the code in operations very carefully. This is known as the "yo yo" problem of deep class hierarchies.

yoyo and super



```
CC aCC = new CC();

aCC.op2();

// op2 of CC called
// op1 of AA called
// op3 of CC called

// in op2 of CC, super.op1() is invoked.

// search starts from BB. op1 not found in BB.

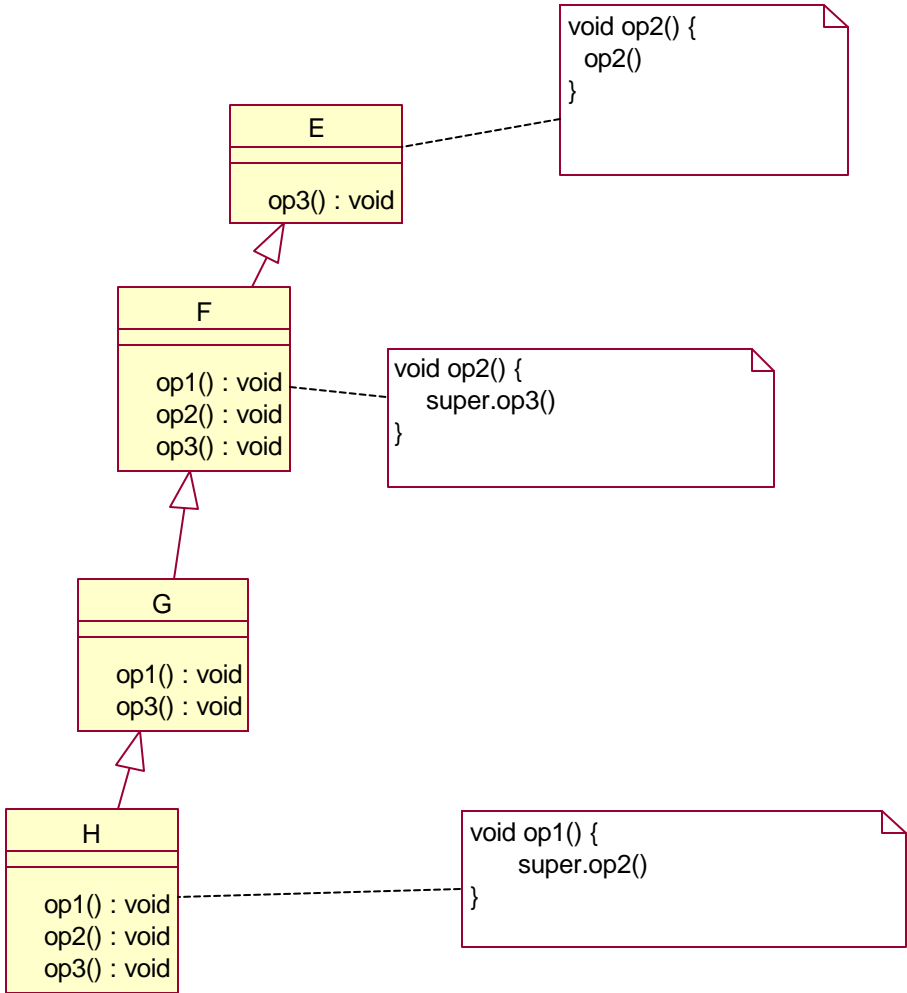
// search for op1 AA. op1 found in AA. op1 of AA invoked.

// op1 of AA calls op3().

// search for op3 starts from AA. Not BB!

// op3 found in AA. op3 of AA is invoked.
```

yoyo and super continued



```
H h = new H();
h.op1()

// op1 of H called
// op2 of F called
// op3 of E called
// op2 of H called
```