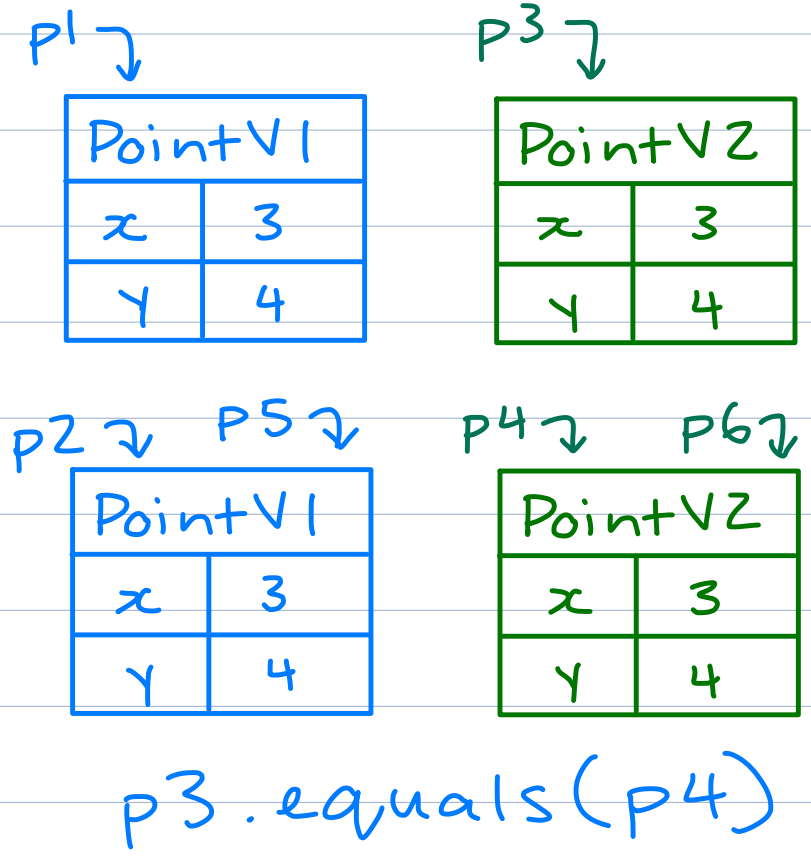
Which ones will fail?

- [ ] a. assertNotSame(p1, p2); ✓
- [ ] b. assertSame(p4, p6); ✓
- [ ] c. assertEquals(p3, p4); ✓
- [ ] d. assertEquals(p2, p5);
- [ ] e. assertSame(p1, p2);



assertSame(a, b)	is	a == b true?
assertNotSame(a, b)	is	a == b false?

assertEqual(a, b)	is	a.equals(b) true?
assertNotEqual(a, b)	is	a.equals(b) false?

Which ones will fail?

- [x] f. assertEquals(p1, p2); ✗
- [ ] g. assertNotSame(p4, p6);
- [x] h. assertNotEquals(p3, p4); ✗
- [ ] i. assertEquals(p5, p6);
- [ ] j. assertEquals(p6, p5);

p1 → PointV1

x	3
y	4

p3 → PointV2

x	3
y	4

p2 → PointV1, p5 →

x	3
y	4

p4 → PointV2, p6 →

x	3
y	4

p3.equals(p4)

assertSame(a, b)	is	a == b true?
assertNotSame(a, b)	is	a == b false?

assertEqual(a, b)	is	a.equals(b) true?
assertNotEqual(a, b)	is	a.equals(b) false?

```
class A extends B {
  A() { }
}
```

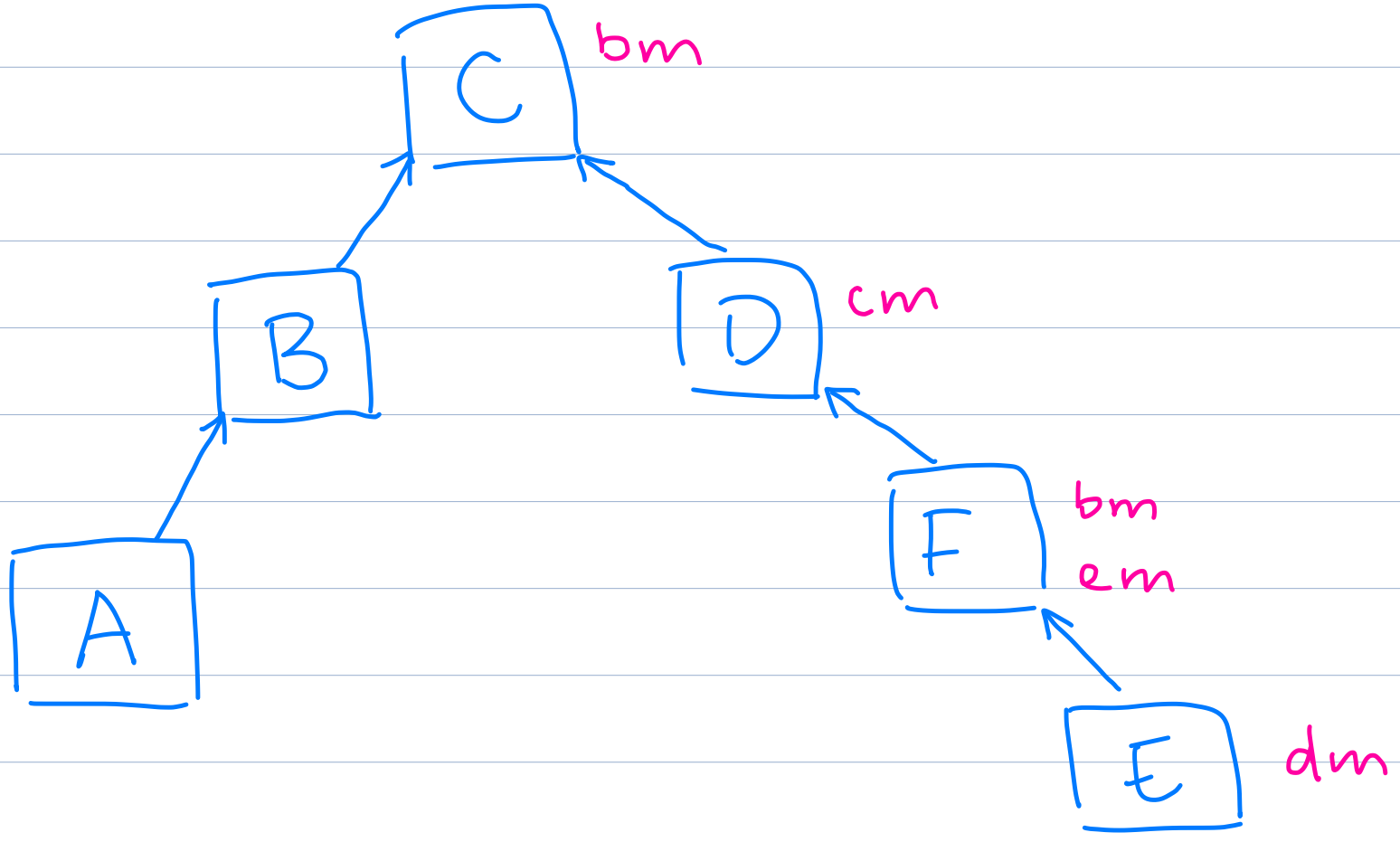
```
class B extends C {
  B() { }
}
```

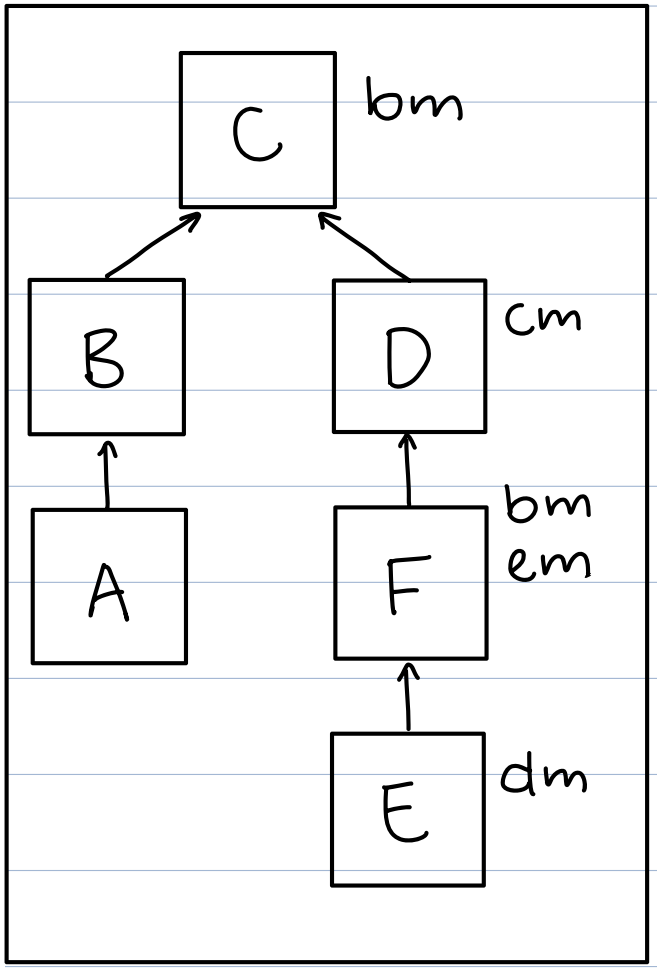
```
class C {
  C() { }
  void bm(){print("C.bm");}
}
```

```
class D extends C {
  D() { }
  void cm(){print("D.cm");}
}
```

```
class F extends D {
  F() { }
  void bm(){print("F.bm");}
  void em(){print("F.em");}
}
```

```
class E extends F {
  E() { }
  void dm(){print("E.dm");}
}
```





Variable Assignment with New Object

Does it compile? X

What is result?

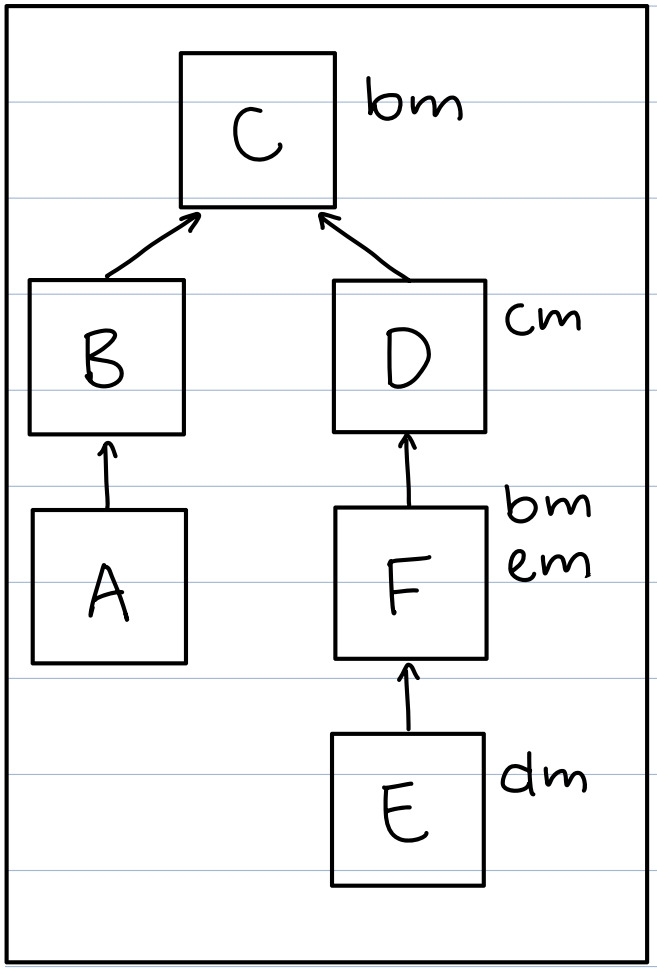
```
D d1 = new C();
```

ST: D

DT: C

Is ~~DT~~ C a descendent of ~~ST~~ D?

Can ~~DT~~ C fulfill expectations of ~~ST~~ D?



Variable Assignment with New Object

Does it compile? ✓

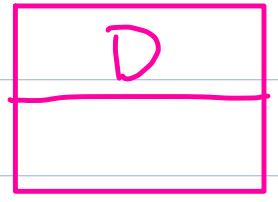
What is result?

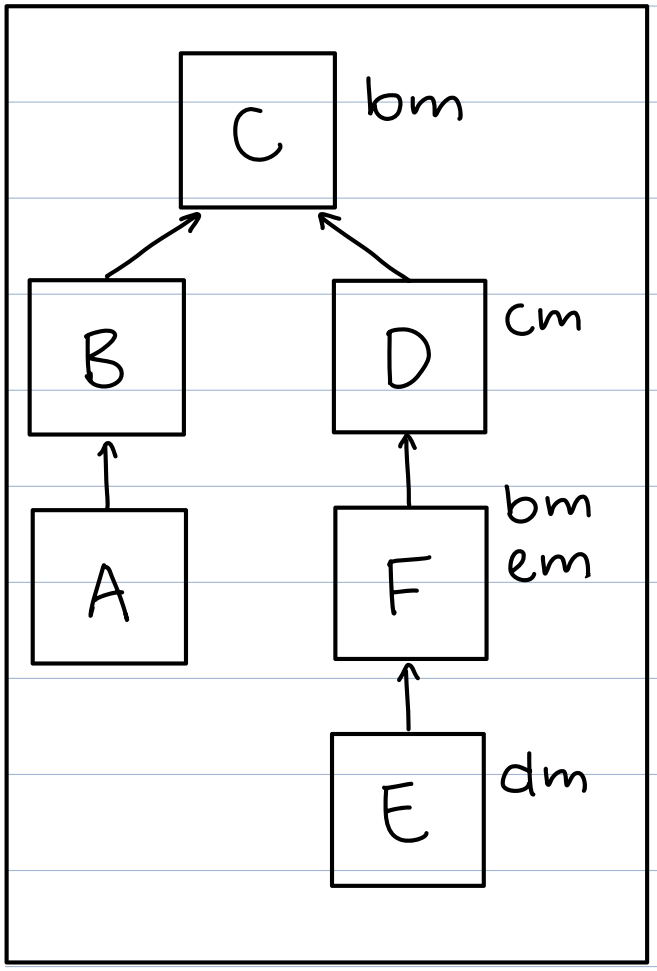
```
C d2 = new D();
```

ST : C

DT : D

C d2 ↘





Method Call

Compilation & Output

Does it compile? ✓

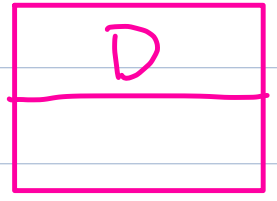
What is output? "C.bm"

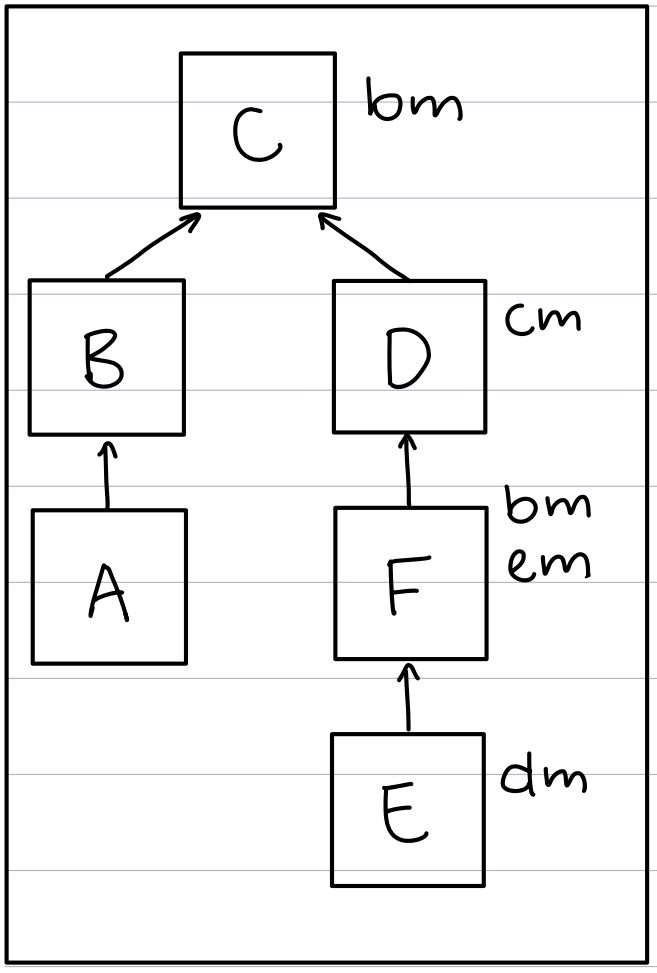
```
d2.bm();
```

ST : C

DT : D

C d2 ↘





Variable Assignment with New Object

Does it compile? ✓

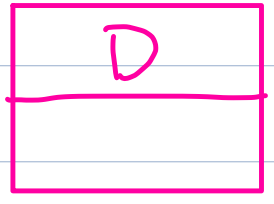
What is result?

```
D e1 = new E();
```

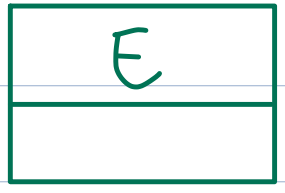
ST : D

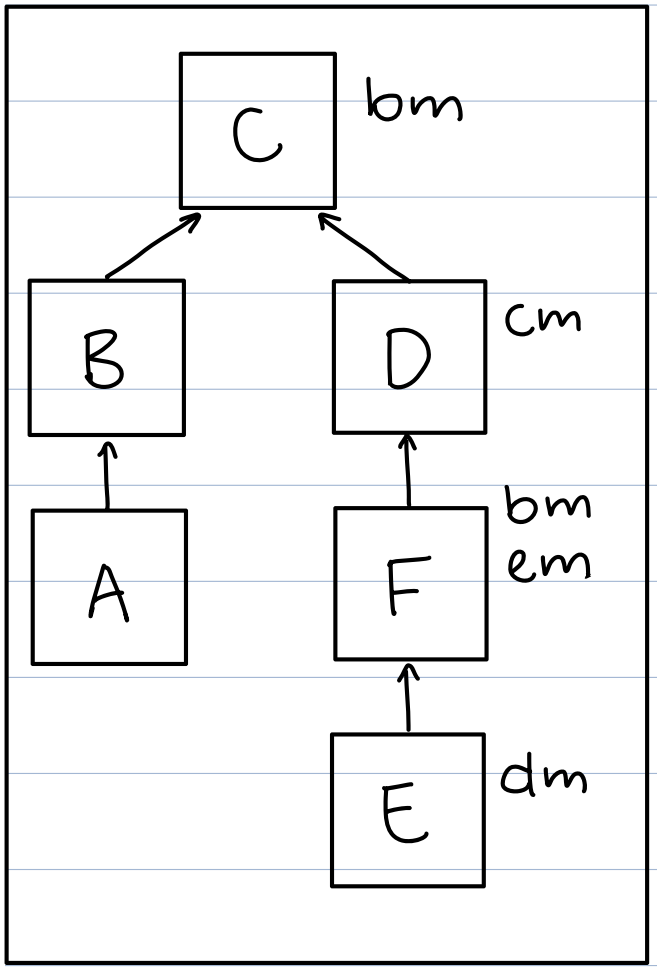
DT : E

C d2 ↘



D e1 ↘





Variable Assignment with Object Reference

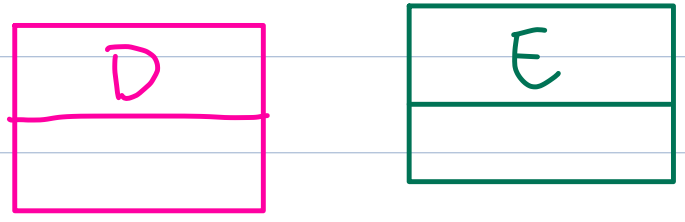
Does it compile? ✓

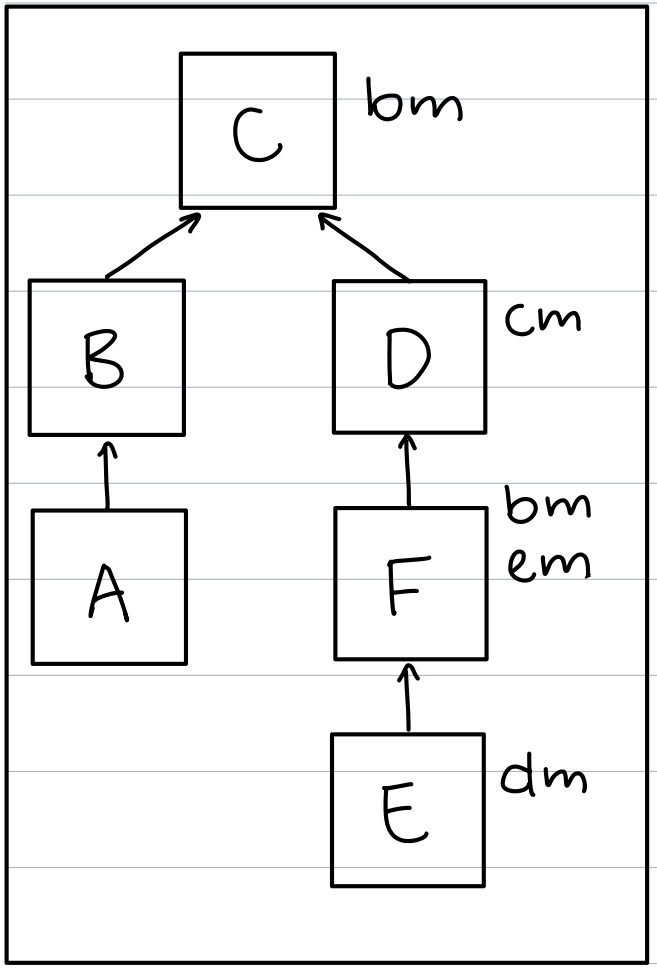
What is result?

```
d2 = e1;
```

ST : C ST : D

D e1 ↘ C d2 ↙





Method Call

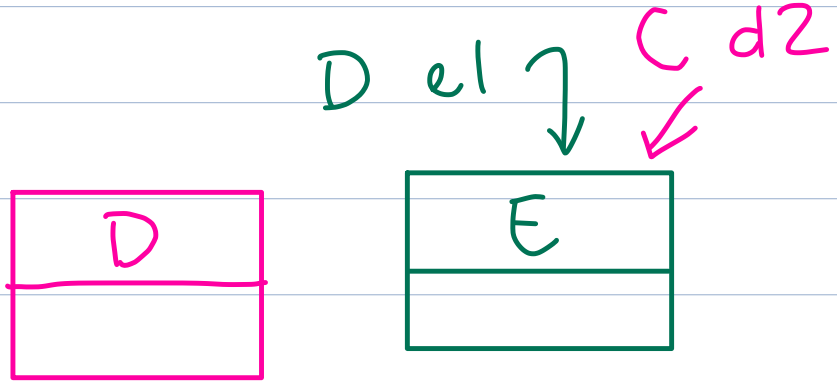
Compilation & Output

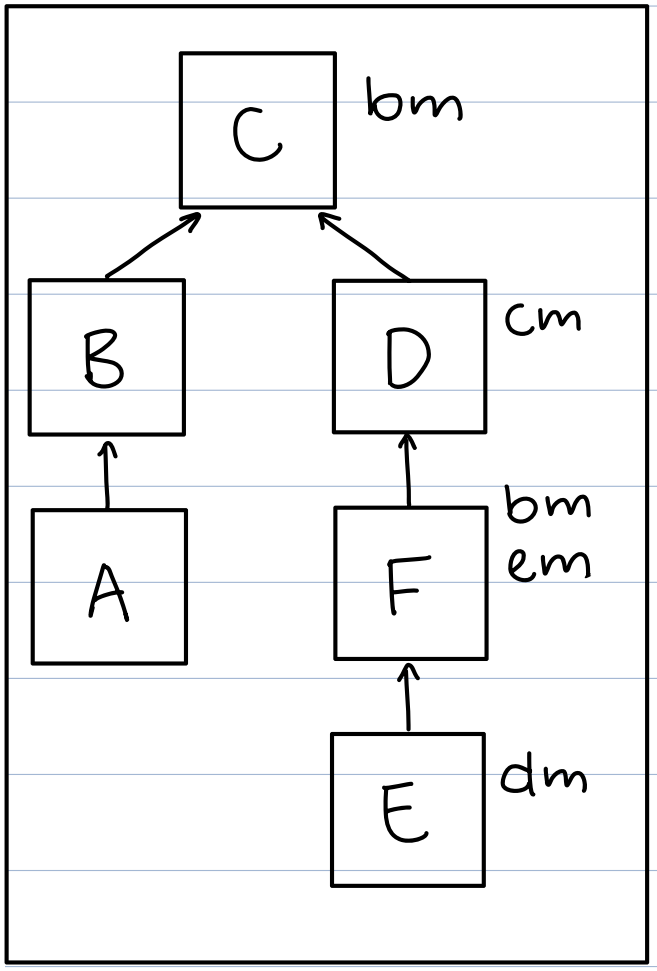
Does it compile? ✓

What is output? F.bm

```
d2.bm();
```

ST: C





Variable Assignment with Object Reference

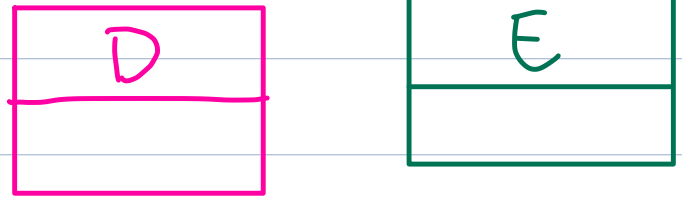
Does it compile? X

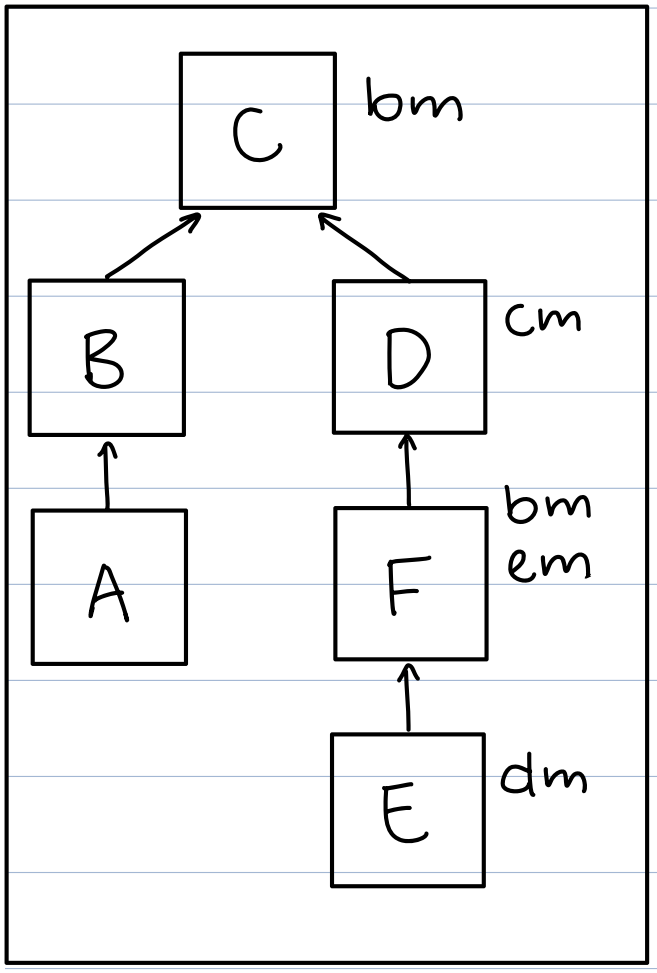
What is result?

```
F f1 = e1;
```

ST : F ST : D

D e1 C d2





Method Call Compilation & Output

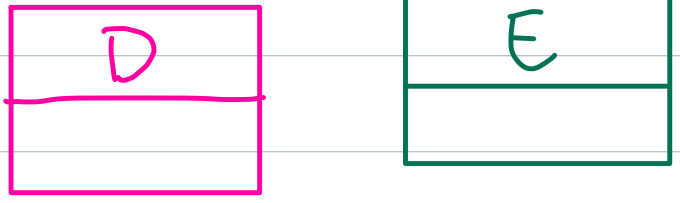
Does it compile? X

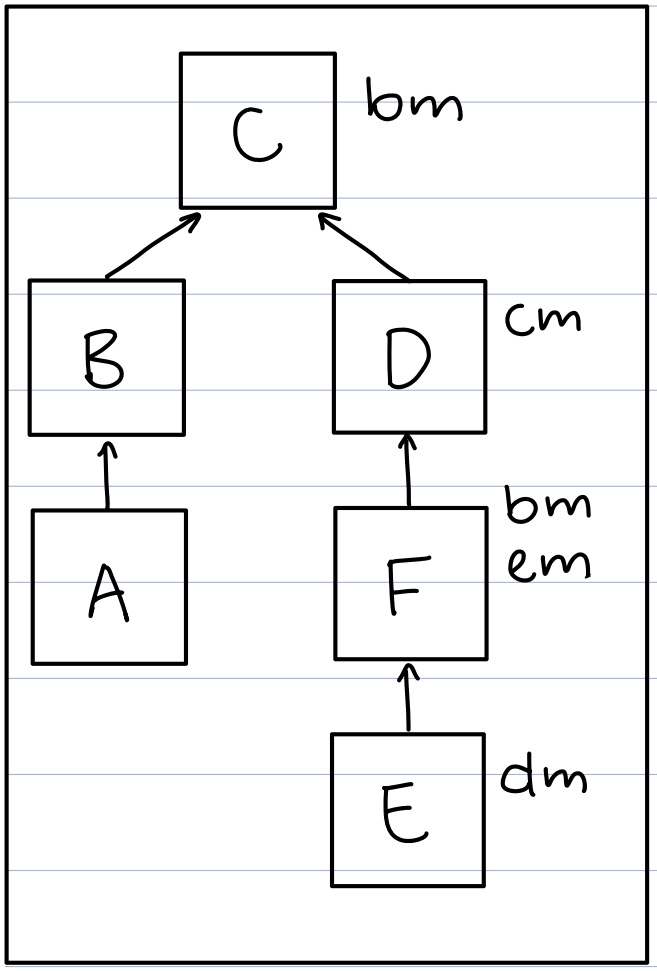
What is output?

```
e1.em();
```

ST: D

D e1 ↘ ↙ C d2





Casting Example 1

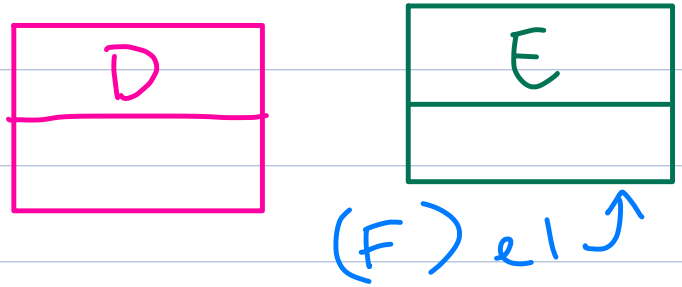
Does it compile? ✓

What is output? F.em

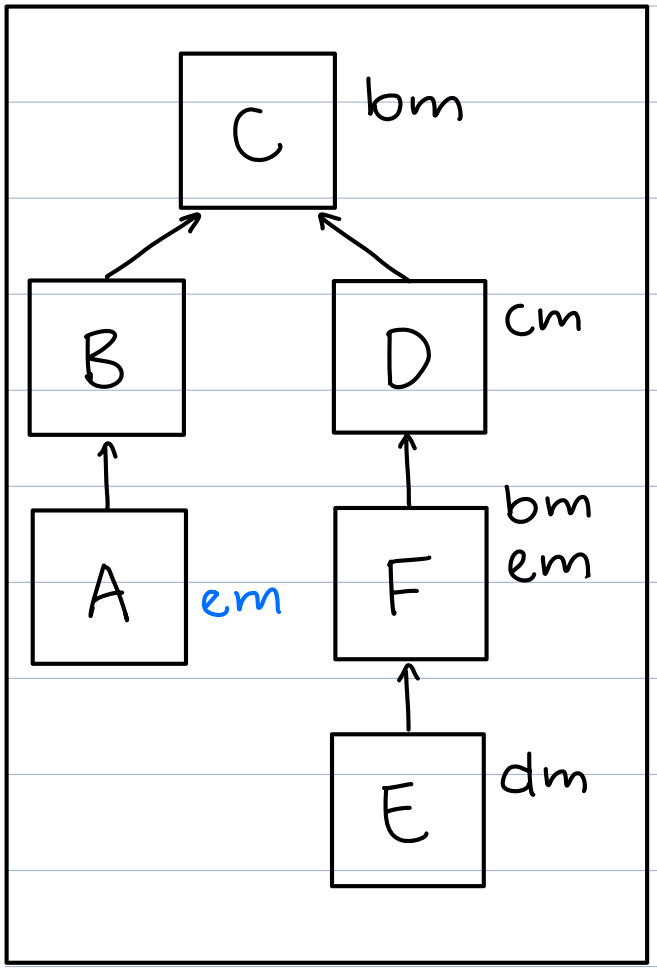
```
((F) e1).em();
```

ST: D

D e1 ↘ ↙ C d2



Note: We can also cast to E



Casting Example 2

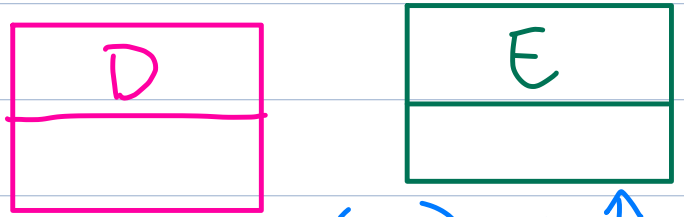
Does it compile? ✗

What is output?

```
((A) e1).em();
```

ST: D

D e1 ↘ ↙ C d2



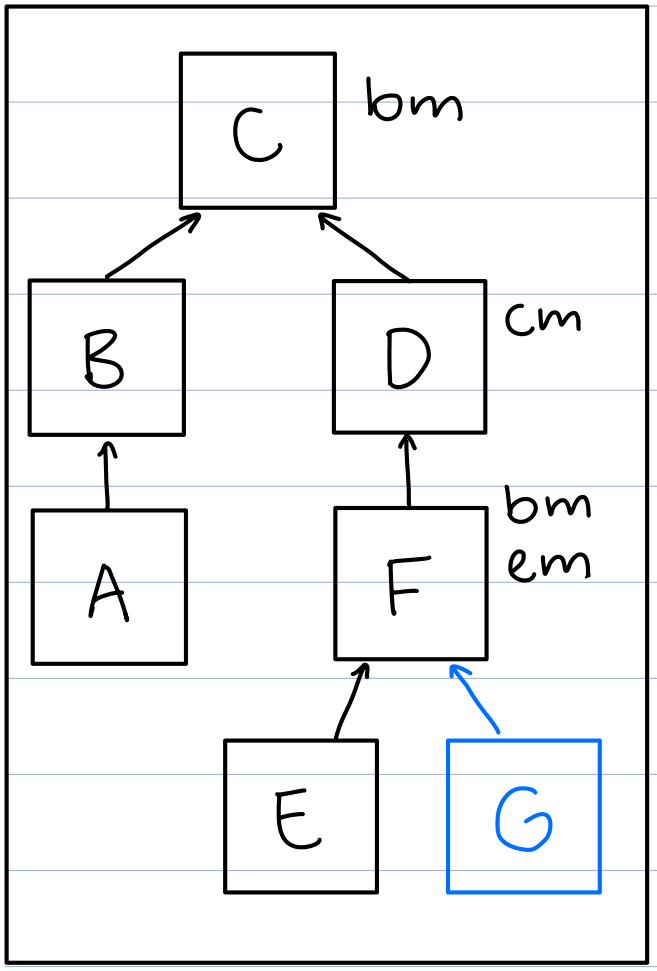
(F) e1 ↗

To make it compile:
upwards cast first,
then downwards cast

```
((A)((C) e1)).em();
```

CCE!

ST: C



Casting
Example 3

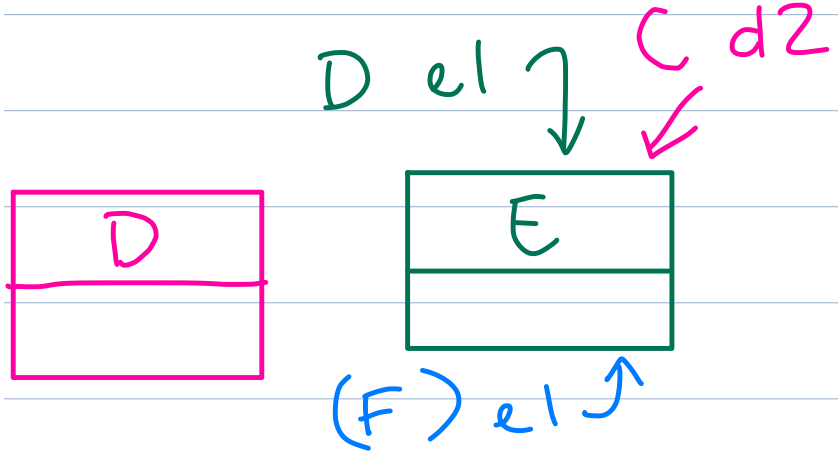
Does it compile? ✓

What is output? ClassCastExc.

```
G g1 = (G) e1
```

ST : D

DT : E



Type Casting

Compilation: Is ST of the variable an ancestor or descendent of the cast type?

↳ if yes, cast compiles (but make sure to check that the rest of the line compiles)

↳ if no, compilation error

Runtime: Is DT of the variable a descendent of the cast type?

↳ if yes, create alias of cast type

↳ if no, ClassCastException

Variable Assignment with New Object

Is DT (RHS) a descendent of ST (LHS)?

↳ if yes, compiles

↳ if no, compilation error

Variable Assignment with Object Reference

Is ST (RHS) a descendent of ST (LHS)?

↳ if yes, compiles

↳ if no, compilation error

Method Calls

Compilation: Is method defined in an ancestor of the ST? (of the context object) (including the ST itself)

↳ if yes, compiles

↳ if no, compilation error

Runtime: Call the method in the closest ancestor of the DT.

C obj = new ---

(C) obj

upward & downward cost.