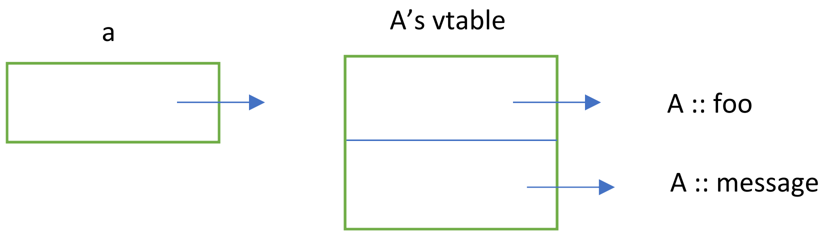
Problem 1:

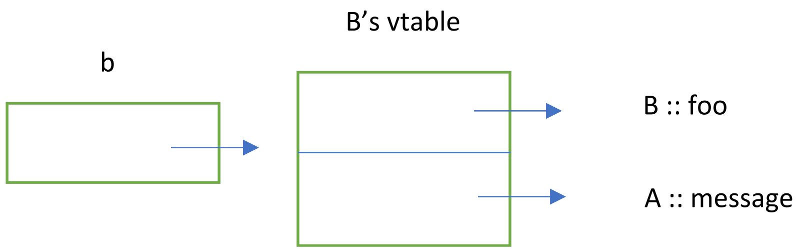
The result of fb.m(fb) will be Foo.m(Foo). Since fb's static class is Foo, the compiler searchers for a method named m in Foo's code. There are two. Since method overloading is resolved statically, the compiler decides the first m method (which takes Foo objects) will be called. Due to the dynamic method lookup and because fb's dynamic class is fb, the m method in Bar that takes Foo objects should be called. Because Bar class inherits the m method that takes Foo objects from Foo class, so the result is Foo.m(Foo).

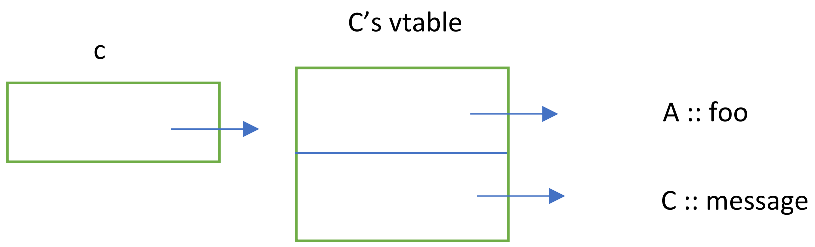
The result of b.m(b) will be Bar.m(Bar). The object "b" is of class "Bar" and the argument, "b", passed into the function call is also of class "Bar". So, the "public void m(Bar b)" function in class "Bar" will be executed and the output will be: Bar.m(Bar).

Problem 2:

Drawing of a, b, c & their vtables (referencing the method as shown on the slides on Canvas with foo and bar classes as well as Figures 10.2 and 10.3 of the class textbook's fourth edition by Scott):


a
A's vtable
A :: foo
A :: message


B's vtable
b
B :: foo
A :: message


C's vtable
c
A :: foo
C :: message