Eric Smrkovsky
CSci 117
12/8/2019
Lab 15

1)

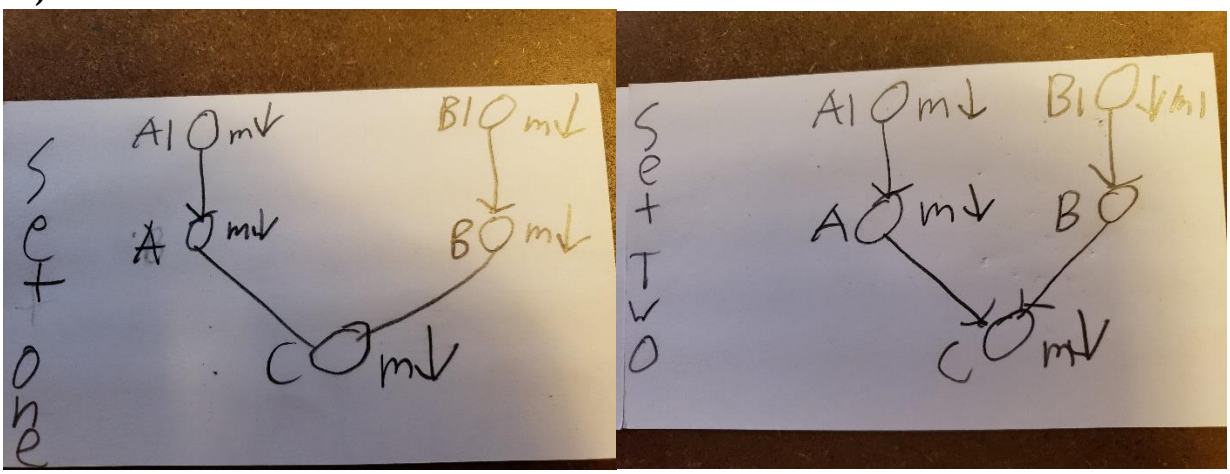
```
//Q1a
fun {New2 Class} Temp in
  class Temp from Class //Temp inherits from Class
    meth init(skip Basic)//this method does nothing
    end
  end
  {New Class init}  //creates an object from the
end                 //class Class and initializes
                    //it with a skip Basic instead
                    //of a value
//Q1b
fun {New2 Class} Temp Init={NewName} in
  class Temp from Class
    meth !Init(skip Basic)
    end
  end
  {New Class Init}
end
```

2)

```
local LoggedAccount Transfer Transfer2 GetBal BatchTransfer in

    proc {Transfer M S}
      transfer(Amt) = M
    in
      {LogObj addentry(transfer(Amt))}
      {Transfer2 transfer(Amt) S}
    end

    proc {Transfer2 M S}
      transfer2(Amt) = M
    in
      (S.balance):=@(S.balance)+Amt
    end

    proc {GetBal M S}
      getBal(Bal) = M
    in
      Bal=@(S.balance)
    end
```

```
  proc {BatchTransfer M S}
    batchTransfer(AmtList)=M
  in
    for A in AmtList do {Transfer transfer(A) S} end
  end

  LoggedAccount=c(attrs:[balance]
    methods:m(transfer:Transfer bal:GetBal bt:BatchTransfer))
 end
end
```

3a)



Yes, set one has errors. A1 and B1 have methods with the same label. Same thing for A and B. Set two is legal with no errors, m is overridden at each level (A1->A->C).

3b) {Obj m()} will output 'C' (the local m() in Class C)
{Obj m1()} will output 'B1' (the local m1() in Class B1 from inheritance)

3c) Method a() will execute which executers method m() and then m1(). When m() executes 'C' is displayed. When m1() executes 'C-m1' is displayed.

3d1)

```
class A1 meth m() {Browse 'A1'} end end
class B1 meth m1() {Browse 'B1'} end end
class A from A1 meth m() {Browse 'A'} end end
class B from B1 end
class C from A B
  meth m() {Browse 'C'} end end
  meth m1() {Browse 'C-m1'} end end
  meth a() A1,m B1,m1 end
end
```

Eric Smrkovsky
CSci 117
12/8/2019
Lab 15

3d2)

Yes, it depends on if the method has been overridden or if it is inherited. If it is inherited, you could call it from the class that inherited it or the superclass that created it. For example, the code below would be a sufficient solution as well, it accesses m1 by inheritance.
meth a() A1,m B,m1 end

4a)

Forwarding example: In the example at the bottom of page 518 two objects are created with the NewF function. Next, the second object calls the setForward method within the class ForwardMixin through inheritance, effectively making itself forward any message that is not understood to the first object. Finally the second object calls the cube method with the message "cube(10 X)", it's class does not understand the message because cube is not a method within object two's class, so cube is forwarded to object one where it is understood.

Delegation example: In the example at the beginning of page 521 two objects are created with the NewD function. Next, the second object calls the setDelegate method within the class DelegateMixin through inheritance, effectively making object one act like a superclass to object two, this is known as a delegation chain. When object two makes a call to the method 'c' it is not defined within object two so the method is retried from object one.

4b)

One of the big differences between forwarding and delegation is how they call themselves. With forwarding, the separate objects each have their own identities and can **forward** messages to and from each other. With delegation, separate objects have a common self which is preserved from an original object and grown into a delegation chain with each call to setDelegate. The hierarchy that is built with the delegation chain is objects delegating its methods to other objects.