


Branch: master [msc\\_project](#) / [OCap](#) / [pony\\_caretaker](#) / [Main.pony](#)[Find file](#) [Copy path](#) shuppyloh updated code

186ebf7 2 minutes ago

0 contributors

136 lines (127 sloc) 6.06 KB

```

1 //The purpose of this snippet is to demonstrate below the use an attenuating object,
2 //caretaker, to mediate access in an OCap system.
3 use collections = "collections"
4 actor Main
5     let env: Env
6     new create(env':Env)=>
7         env = env'
8         env.out.print("---Initial Conditions---")
9         let alice: SimpleObj ref = SimpleObj.create(env,"alice")
10        let bob: SimpleObj ref = SimpleObj.create(env,"bob")
11        let carol: SimpleObj ref = SimpleObj.create(env,"carol")
12        let diane: SimpleObj ref = SimpleObj.create(env,"diane")
13        try
14            //initial conditions
15            alice.recCap("bob",bob)
16            alice.recCap("carol",carol)
17            carol.recCap("diane",diane)
18            diane.sendProp("diane_prop1","true","diane")
19            env.out.print("---Initial Conditions Completed---")
20
21            //Alice passing a caretaker for Carol, to Bob for Bob's use
22            alice.createCaretaker("carol-CT","carol") //carol-CT caretaker created
23            alice.sendCap("carol-CT","bob") //alice sends carol-CT to bob
24            //Bob sending his own capability to Carol
25            bob.sendCap("bob","carol-CT")
26            //Carol sending Diane's capability to Bob
27            carol.sendCap("diane","bob")
28            //Bob tells sets prop1 in Carol to be true
29            bob.sendProp("carol_prop1","true","carol-CT") //bob sends property (prop1 = true) to carol-CT
30            env.out.print("MAIN:carol_prop1 is "+carol.getProp("carol_prop1")) //this carol's prop1 should return true
31
32            //POST-LOCK
33
34            //Alice changes lock of Carol-CT
35            alice.changeLock(true,"carol-CT-lock") //alice locks carol-CT
36            //Bob tries to change prop1=false on carol-CT and the lock should prevent him from doing so
37            bob.sendProp("carol_prop1","false","carol-CT") //bob tries to change prop1 = false to carol-CT
38            env.out.print("MAIN:carol_prop1 is "+carol.getProp("carol_prop1")) //this carol's prop1 should return true
39            //Bob tries to change prop1=false on diane and will succeed because nothing is preventing him from doing so
40            bob.sendProp("diane_prop1","false","diane") //bob tries to change prop1 = false to diane
41            env.out.print("MAIN:diane_prop1 is "+diane.getProp("diane_prop1")) //because caretaker is locked, should return true
42
43        end
44
45    class Lock
46        var _state: Bool val
47        new ref create()=>
48            _state = false
49        fun ref unlock()=>
50            _state = false
51        fun ref lock()=>
52            _state = true
53        fun box state():Bool val=>

```

```

54     _state
55
56 class Caretaker
57     let _target: (Caretaker ref|SimpleObj ref)
58     var _lock: Lock ref
59     new ref create(target':(Caretaker ref|SimpleObj ref), lock':Lock ref)=>
60         _target = target'
61         _lock = lock'
62     fun box _locked():Bool val=>
63         _lock.state()
64     fun box getProp(id:String val):String val?=>
65         try
66             if _locked() is false then _target.getProp(id) else error end
67         else error end
68     fun ref sendProp(id:String val,prop:String val,rec: String val)?=>
69         try
70             if _locked() is false then _target.sendProp(id,prop,rec) else error end
71         else error end
72     fun ref recProp(id:String val,prop:String val)=>
73         if _locked() is false then _target.recProp(id,prop) end
74     fun ref getCap(id:String val): (SimpleObj ref|Lock ref|Caretaker ref)?=>
75         try
76             if _locked() is false then _target.getCap(id) else error end
77         else error end
78     fun ref sendCap(id:String val, rec:String val)?=>
79         try
80             if _locked() is false then _target.sendCap(id,rec) end
81         else error end
82     fun ref recCap(id:String val, cap':(SimpleObj ref|Lock ref|Caretaker ref))=>
83         if _locked() is false then _target.recCap(id,cap') end
84     fun ref delCap(id:String val)?=>
85         try
86             if _locked() is false then _target.delCap(id) end
87         else error end
88     fun ref createCareT(id:String val,target:String val):Caretaker ref?=>
89         try
90             if _locked() is false then _target.createCareT(id,target) else error end
91         else error end
92
93 class SimpleObj
94     let env: Env
95     let name: String
96     let _caps: collections.Map[String val, (SimpleObj ref|Lock ref|Caretaker ref)] = _caps.create()
97     let _props: collections.Map[String val, String val] = _props.create()
98
99     new ref create(env':Env, name':String)=>
100         env = env'; name = name'
101         _caps(name)=this
102     fun ref changelock(lock:Bool val,rec: String val)?=>
103         try if lock is true then (getCap(rec) as Lock ref).lock()
104             else (getCap(rec) as Lock ref).unlock() end
105         env.out.print(name+": changing lock of "+rec+" to "+lock.string())
106         else error end
107     fun box getProp(id:String val):String val ?=>
108         try _props(id) else error end
109     fun ref sendProp(id:String val,prop:String val,rec: String val)?=>
110         env.out.print(name+":sending (" +id+" as "+prop+") to "+rec)
111         try (getCap(rec) as (Caretaker ref|SimpleObj ref)).recProp(id,prop) else error end
112     fun ref recProp(id:String val,prop:String val) =>
113         env.out.print(name+": "+id+" changed to "+prop)
114         _props(id)=prop
115     fun ref getCap(id:String val): (SimpleObj ref|Lock ref|Caretaker ref)?=>
116         try _caps(id) else error end
117     fun ref sendCap(id:String val, rec:String val)?=>
118         env.out.print(name+":sending capability of "+id+" to "+rec)
119         try (getCap(rec) as (Caretaker ref|SimpleObj ref)).recCap(id, getCap(id)) else error end

```

```
120 fun ref recCap(id:String val, cap':(SimpleObj ref|Lock ref|Caretaker ref))=>
121   _caps(id) = cap'
122   env.out.print(name+":received capability of "+id)
123 fun ref delCap(id:String val) ?=>
124   try _caps.remove(id) else error end
125 fun ref createCareT(id:String val,target':String val):Caretaker ref?=>
126   env.out.print(name+":creating caretaker "+id+" for "+target')
127   try
128     let cap = (getCap(target') as (Caretaker ref|SimpleObj ref))
129     let lockname: String val = id+"-lock"
130     let lock:Lock ref = Lock.create()
131     let caretaker:Caretaker ref = Caretaker.create(cap,lock)
132     recCap(lockname, lock)
133     recCap(id, caretaker)
134     caretaker
135   else error end
```