shuppyloh / msc_project

Branch: master msc_project / OCap / pony_membrane / Main.pony Find file Copy path

shuppyloh updated code 186ebf7 4 minutes ago

0 contributors

158 lines (149 sloc) 7.06 KB

```
//The purpose of this snippet is to demonstrate below the use a DEEP attenuating object,
//membrane, to mediate access in an OCap system.
use collections = "collections"
actor Main
    let env: Env
    new create(env':Env)=>
        env = env'
        env.out.print("---Initial Conditions---")
        let alice: SimpleObj ref = SimpleObj.create(env,"alice")
        let bob: SimpleObj ref = SimpleObj.create(env,"bob")
        let carol: SimpleObj ref = SimpleObj.create(env,"carol")
        let diane: SimpleObj ref = SimpleObj.create(env,"diane")
        try
        //initial conditions
        alice.recCap("bob",bob)
        alice.recCap("carol",carol)
        carol.recCap("diane",diane)
        diane.sendProp("diane_prop1","true","diane")
        env.out.print("---Initial Conditions Completed---")

        //Alice passing a caretaker for Carol, to Bob for Bob's use
        alice.createMemb("carol-M","carol") //carol-M caretaker created
        alice.sendCap("carol-M","bob") //alice sends carol-M to bob
        //Bob sending his own capability to Carol
        bob.sendCap("bob","carol-M")
        //Carol sending Diane's capability to Bob
        carol.sendCap("diane","bob-M")
        //Bob tells sets prop1 in Carol to be true
        bob.sendProp("carol_prop1","true","carol-M") //bob sends property (prop1 = true) to carol-M
        env.out.print("MAIN:carol_prop1 is "+carol.getProp("carol_prop1")) //this carol's prop1 should return true

        //POST-LOCK

        //Alice changes lock of Carol-M
        alice.changelock_all(true,"carol-M-lock") //alice locks carol-M
        //Bob tries to change prop1=false on carol-M and the lock should prevent him from doing so
        bob.sendProp("carol_prop1","false","carol-M") //bob tries to change prop1 = false to carol-M
        env.out.print("MAIN:carol_prop1 is "+carol.getProp("carol_prop1")) //this carol's prop1 should return true
        //Bob tries to change prop1=false on diane and will fail because membrane will stop this
        bob.sendProp("diane_prop1","false","diane-M") //bob tries to change prop1 = false to diane
        env.out.print("MAIN:diane_prop1 is "+diane.getProp("diane_prop1")) //because membrane is locked, should return true

        end

class Lock
    var _state: Bool val
    let _children: collections.Map[String val, Lock ref] = _children.create()
    new ref create()=>
        _state = false
    fun ref addchild(id': String val, lock': Lock ref)=>
        _children(id')=lock'
    fun ref unlock()=>
        _state = false
```

```
    fun ref lock()=>
        _state = true
    fun ref unlockall()=>
        for child in _children.values() do child.unlockall() end
        _state = false
    fun ref lockall()=>
        for child in _children.values() do child.lockall() end
        _state = true
    fun box state():Bool val=>
        _state

class Membrane
    let _target: (SimpleObj ref|Membrane ref)
    let _lock: Lock ref
    let _children: collections.Map[String val, Membrane ref] = _children.create()
    new ref create(target':(SimpleObj ref|Membrane ref), lock':Lock ref)=>
        _target = target'
        _lock = lock'
    fun box _locked():Bool val=>
        _lock.state()
    fun box getProp(id:String val):String val?=>
        try
            if _locked() is false then _target.getProp(id) else error end
        else error end
    fun ref sendProp(id:String val,prop:String val,rec: String val)?=>
        try
            if _locked() is false then _target.sendProp(id,prop,rec) else error end
        else error end
    fun ref recProp(id:String val,prop:String val)=>
        if _locked() is false then _target.recProp(id,prop) end
    fun ref getCap(id:String val): (SimpleObj ref|Lock ref|Membrane ref)?=>
        try
            if _locked() is false then _target.getCap(id) else error end
        else error end
    fun ref sendCap(id:String val, rec:String val)?=>
        try
            if _locked() is false then _target.sendCap(id,rec) end
        else error end
    fun ref recCap(id:String val, cap':(SimpleObj ref|Lock ref|Membrane ref))?=> //wrap capability parameter in method
    try
        if _locked() is false then
        let newlock:Lock ref = Lock.create()
        _lock.addchild(id+"-lock",newlock)
        let newMemb:Membrane ref = Membrane.create((cap' as (SimpleObj ref|Membrane ref)),newlock)
        _children(id) = newMemb
        _target.recCap(id+"-M",newMemb) end
    else error end
    fun ref delCap(id:String val)?=>
        try
            if _locked() is false then _target.delCap(id) end
        else error end
    fun ref createMemb(id:String val,target:String val):Membrane ref?=>
        try
            if _locked() is false then _target.createMemb(id,target) else error end
        else error end

class SimpleObj
    let env: Env
    let name: String
    let _caps: collections.Map[String val, (SimpleObj ref|Lock ref|Membrane ref)] = _caps.create()
    let _props: collections.Map[String val, String val] = _props.create()

    new ref create(env':Env, name':String)=>
        env = env'; name = name'
        _caps(name) = this
    fun ref changelock(lock:Bool val,rec: String val)?=>
```

```
        try if lock is true then (getCap(rec) as Lock ref).lock()
        else (getCap(rec) as Lock ref).unlock() end
        env.out.print(name+": changing single lock of "+rec+" to "+lock.string())
        else error end
    fun ref changelock_all(lock:Bool val,rec: String val)?=>
        try if lock is true then (getCap(rec) as Lock ref).lockall()
        else (getCap(rec) as Lock ref).unlockall() end
        env.out.print(name+": changing entire membrane lock of "+rec+" to "+lock.string())
        else error end
    fun box getProp(id:String val):String val ?=>
        try _props(id) else error end
    fun ref sendProp(id:String val,prop:String val,rec: String val)?=>
        env.out.print(name+":sending ("+id+" as "+prop+") to "+rec)
        try (getCap(rec) as (Membrane ref|SimpleObj ref)).recProp(id,prop) else error end
    fun ref recProp(id:String val,prop:String val) =>
        env.out.print(name+":"+id+" changed to "+prop)
        _props(id)=prop
    fun ref getCap(id:String val): (SimpleObj ref|Lock ref|Membrane ref)?=>
        try _caps(id) else error end
    fun ref sendCap(id:String val, rec:String val)?=>
        env.out.print(name+":sending capability of "+id+" to "+rec)
        try (getCap(rec) as (Membrane ref|SimpleObj ref)).recCap(id, getCap(id)) else error end
    fun ref recCap(id:String val, cap':(SimpleObj ref|Lock ref|Membrane ref))=>
        _caps(id) = cap'
        env.out.print(name+":received capability of "+id)
    fun ref delCap(id:String val) ?=>
        try _caps.remove(id) else error end
    fun ref createMemb(id:String val,target':String val):Membrane ref?=>
        env.out.print(name+":creating membrane "+id+" for "+target')
        try
            let cap = (getCap(target') as (Membrane ref|SimpleObj ref))
            let lockname:String val = id+"-lock"
            let lock:Lock ref = Lock.create()
            let membrane:Membrane ref = Membrane.create(cap,lock)
            recCap(lockname, lock)
            recCap(id, membrane)
            membrane
        else error end
```