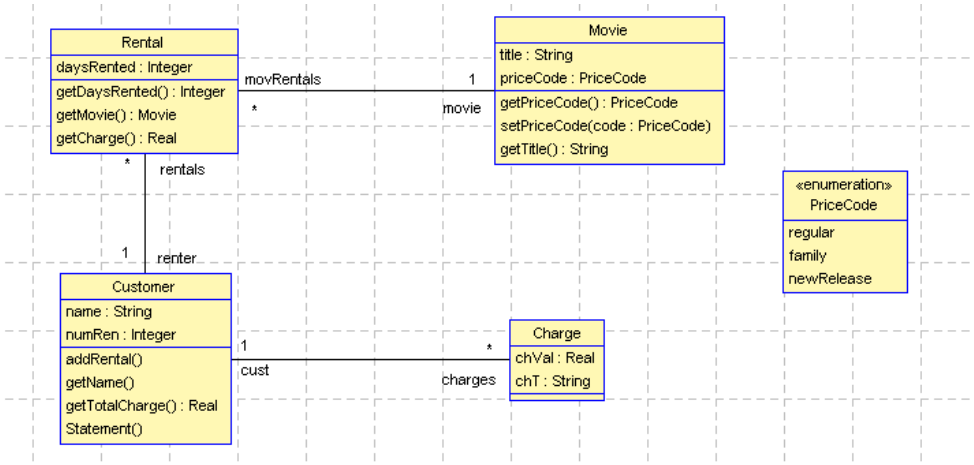
Bryce Lehnen

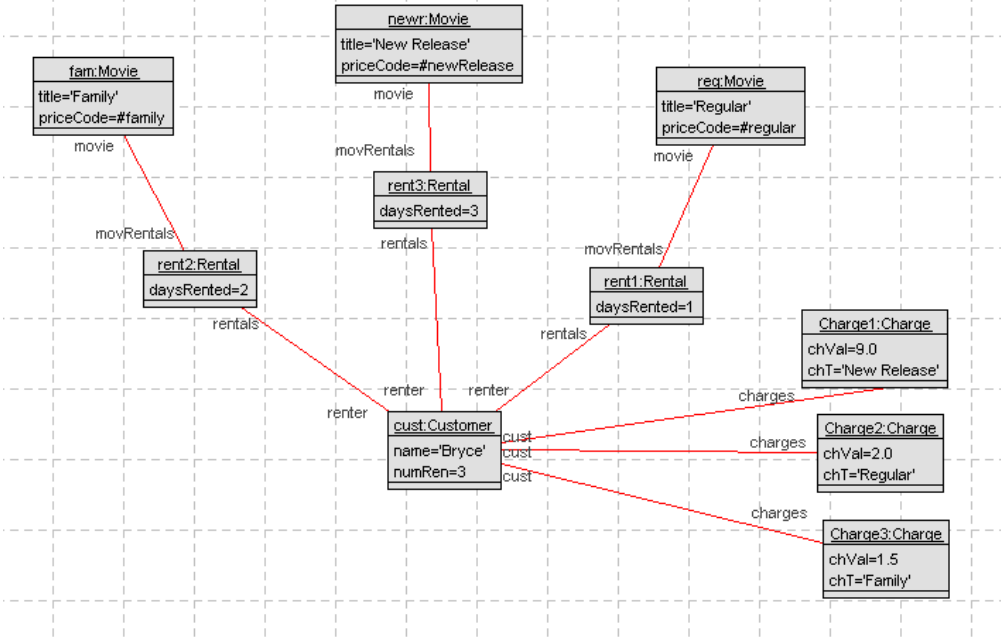
```
-- This is a USE model that has embedded SOIL operations in it
-- Question 1

model MovieRental

enum PriceCode {regular, family, newRelease}

--classes

class Customer
attributes
   name:String
   numRen:Integer

operations
   addRental()
      begin
      end

   getName()

   getTotalCharge():Real
      begin
         declare total:Real, aCharge:Charge, m:Movie, ch:Real;
         total:=0;
         for ren in self.rentals do
            ch:=ren.getCharge();
            total:=total + ch;
         end;
         result:=total;
      end

      Statement()
         begin
            declare aCharge:Charge, sm:Movie, ch:Real, t:String;
            self.numRen:=self.rentals->size();
            for ren in self.rentals do
               ch:=ren.getCharge();
               sm:=ren.getMovie();
               t:=sm.getTitle();
               aCharge:= new Charge;
               aCharge.chVal:=ch;
               aCharge.chT:=t;
               insert(self,aCharge) into customerCharges
            end;
         end
end
```

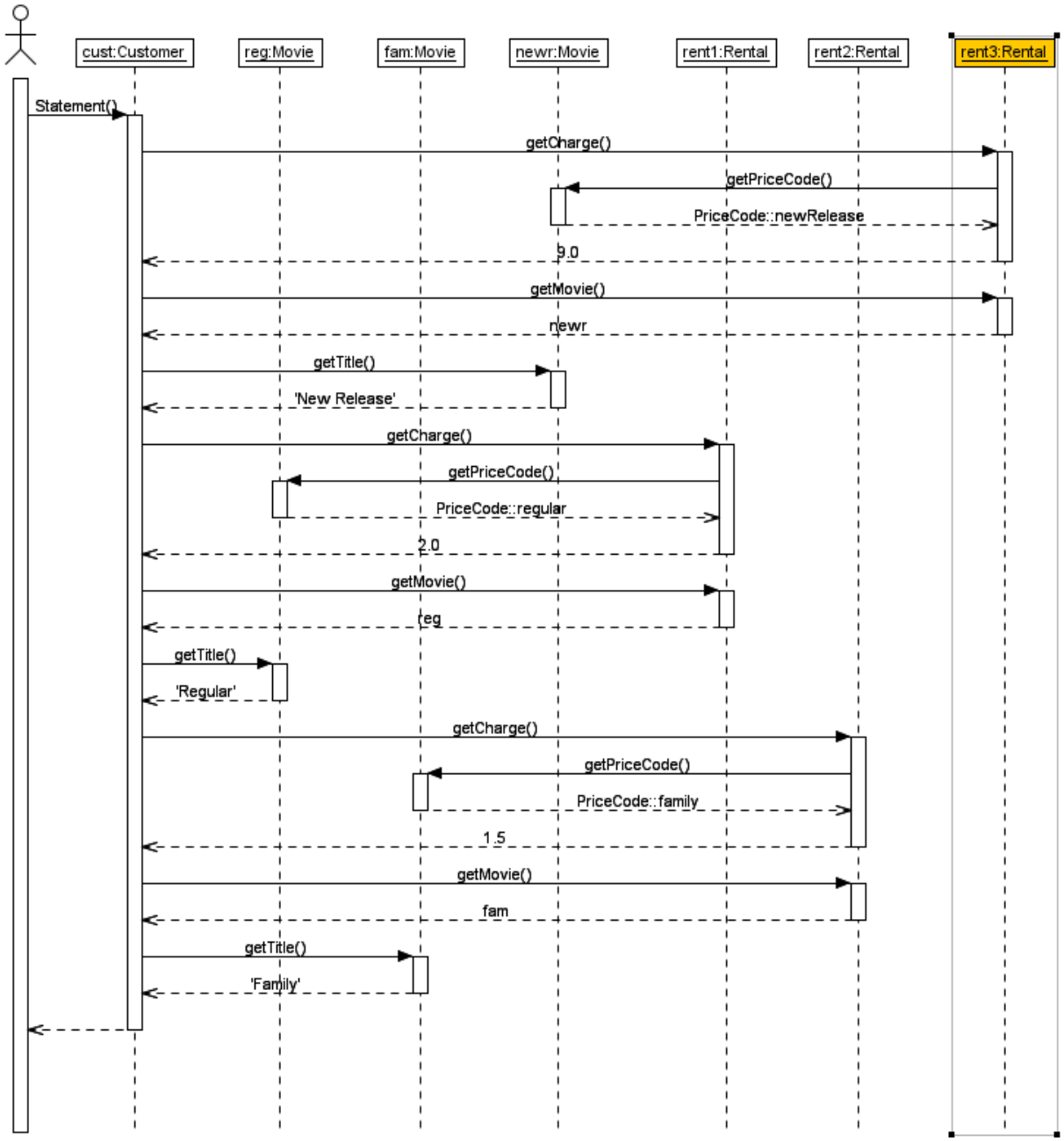
```
class Rental
attributes
   daysRented:Integer

operations
   getDaysRented():Integer
      begin
         result := self.daysRented;
      end

   getMovie(): Movie
      begin
         result := self.movie;
      end

   getCharge():Real
      begin
         declare wrkCh:Real, m:Movie, pc:PriceCode, dy:Integer;
         wrkCh:=0;
         m:=self.movie;
         dy:=self.daysRented;
         pc:=m.getPriceCode();
         if pc=PriceCode::regular then
            wrkCh:=2.0;
            if dy > 2 then
               wrkCh:=wrkCh + (dy - 2) * 1.5;
            end;
         end;
         if pc=PriceCode::family then
            wrkCh:=1.5;
            if dy > 3 then
               wrkCh:=wrkCh + (dy - 3) * 1.5;
            end;
         end;
         if pc=PriceCode::newRelease then
            wrkCh:=dy * 3.0;
         end;
         result:=wrkCh;
      end
end

class Movie
attributes
   title:String
   priceCode:PriceCode

operations
   getPriceCode():PriceCode
```

```
        begin
            result := self.priceCode;
        end

    setPriceCode(code:PriceCode)
        begin
            self.priceCode := code;
        end

    getTitle():String
        begin
            result := self.title;
        end
end

class Charge
attributes
    chVal:Real
    chT: String
operations
end

--associations

association custRentals between
    Customer [1] role renter
    Rental [0..*] role rentals
end

association movRental between
    Rental [0..*] role movRentals
    Movie [1] role movie
end

association customerCharges between
    Customer [1] role cust
    Charge [0..*] role charges
end

--constraints

constraints
--Example constraints
--You may remove these constraints in your design. They are here
--just as examples.

context Customer
    inv maxRental:numRen <= 10
    inv agreement:rentals->size = numRen
```

```
    inv rentals:rentals->notEmpty
    inv daysRented:rentals->select(daysRented > 3)->notEmpty

-- initiate
!create cust:Customer
!create reg:Movie
!create fam:Movie
!create newr:Movie
!create rent1:Rental
!create rent2:Rental
!create rent3:Rental

-- set values
!set cust.name := 'Bryce'
!set cust.numRen := 0

!set reg.title := 'Regular'
!set reg.priceCode := PriceCode::regular
!set fam.title := 'Family'
!set fam.priceCode := PriceCode::family
!set newr.title := 'New Release'
!set newr.priceCode := PriceCode::newRelease

!set rent1.daysRented := 1
!set rent2.daysRented := 2
!set rent3.daysRented := 3

-- insert associations
!insert (cust, rent1) into custRentals
!insert (cust, rent2) into custRentals
!insert (cust, rent3) into custRentals

!insert (rent1, reg) into movRental
!insert (rent2, fam) into movRental
!insert (rent3, newr) into movRental

!cust.Statement()
```


Rental
daysRented : Integer
getDaysRented() : Integer
getMovie() : Movie
getCharge() : Real
movRentals
*
1
movie
Movie
title : String
priceCode : PriceCode
getPriceCode() : PriceCode
setPriceCode(code : PriceCode)
getTitle() : String
*
rentals
1
renter
«enumeration»
PriceCode
regular
family
newRelease
Customer
name : String
numRen : Integer
addRental()
getName()
getTotalCharge() : Real
Statement()
1
cust
*
charges
Charge
chVal : Real
chT : String


fam:Movie
title='Family'
priceCode=#family
movie
newr:Movie
title='New Release'
priceCode=#newRelease
movie
req:Movie
title='Regular'
priceCode=#regular
movie
movRentals
rent2:Rental
daysRented=2
movRentals
rent3:Rental
daysRented=3
movRentals
rent1:Rental
daysRented=1
rentals
rentals
rentals
renter
renter
renter
cust:Customer
name='Bryce'
numRen=3
cust
cust
cust
Charge1:Charge
chVal=9.0
chT='New Release'
charges
Charge2:Charge
chVal=2.0
chT='Regular'
charges
Charge3:Charge
chVal=1.5
chT='Family'
charges



-- This is a USE model that has embedded SOIL operations in it
-- Question 2

model ModifiedMovie

enum PriceCode {regular, family, newRelease}

--classes

class Customer

```
attributes
   name:String
   numRen:Integer

operations
   addRental()
      begin
      end

   getName()

   getAmount(aRen:Rental):Real
      begin
         declare wrkCh:Real, m:Movie, pc:PriceCode,dy:Integer;
         m:=aRen.getMovie();
         dy:=aRen.getDaysRented();
         pc:=m.getPriceCode();
         wrkCh:=0;
         if pc=PriceCode::regular then
            wrkCh:=2.0;
            if dy > 2 then
               wrkCh:=wrkCh + (dy - 2) * 1.5;
            end;
         end;
         if pc=PriceCode::family then
            wrkCh:=1.5;
            if dy > 3 then
               wrkCh:=wrkCh + (dy - 3) * 1.5;
            end;
         end;
         if pc=PriceCode::newRelease then
            wrkCh:=dy * 3.0;
         end;
         result:=wrkCh;
      end

      Statement()
         begin
            declare aCharge:Charge, sm:Movie, ch:Real, t:String;
            self.numRen:=self.rentals->size();
            for ren in self.rentals do
               /*ch:=ren.getCharge();*/
               sm:=ren.getMovie();
               t:=sm.getTitle();
               aCharge:= new Charge;
               aCharge.chVal:=ch;
               aCharge.chT:=t;
               insert(self,aCharge) into customerCharges
            end;
```

```
        end
end

class Rental
attributes
    daysRented:Integer

operations
    getDaysRented():Integer
        begin
            result := self.daysRented;
        end

    getMovie(): Movie
        begin
            result := self.movie;
        end
end

class Movie
attributes
    title:String
    priceCode:PriceCode

operations
    getPriceCode():PriceCode
        begin
            result := self.priceCode;
        end

    setPriceCode(code:PriceCode)
        begin
            self.priceCode := code;
        end

    getTitle():String
        begin
            result := self.title;
        end
end

class Charge
attributes
    chVal:Real
    chT: String
operations
end

--associations
```

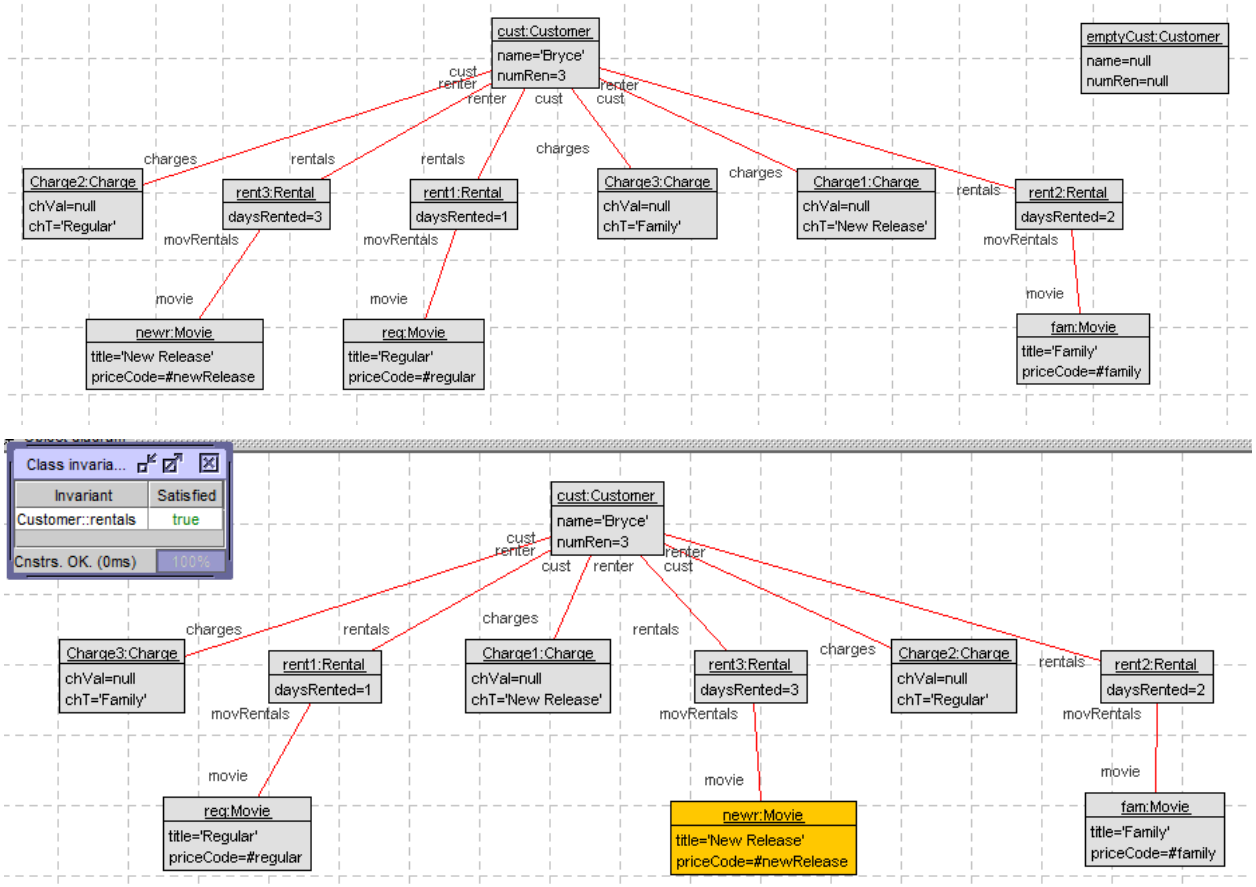
```
association custRentals between
   Customer [1] role renter
   Rental [0..*] role rentals
end

association movRental between
   Rental [0..*] role movRentals
   Movie [1] role movie
end

association customerCharges between
   Customer [1] role cust
   Charge [0..*] role charges
end

--constraints

constraints
--Example constraints
--You may remove these constraints in your design. They are here
--just as examples.

context Customer
   inv rentals:rentals->notEmpty

-- initiate
!create cust:Customer
/*!create emptyCust:Customer*/
!create reg:Movie
!create fam:Movie
!create newr:Movie
!create rent1:Rental
!create rent2:Rental
!create rent3:Rental

-- set values
!set cust.name := 'Bryce'
!set cust.numRen := 0

!set reg.title := 'Regular'
!set reg.priceCode := PriceCode::regular
!set fam.title := 'Family'
!set fam.priceCode := PriceCode::family
!set newr.title := 'New Release'
!set newr.priceCode := PriceCode::newRelease

!set rent1.daysRented := 1
!set rent2.daysRented := 2
```

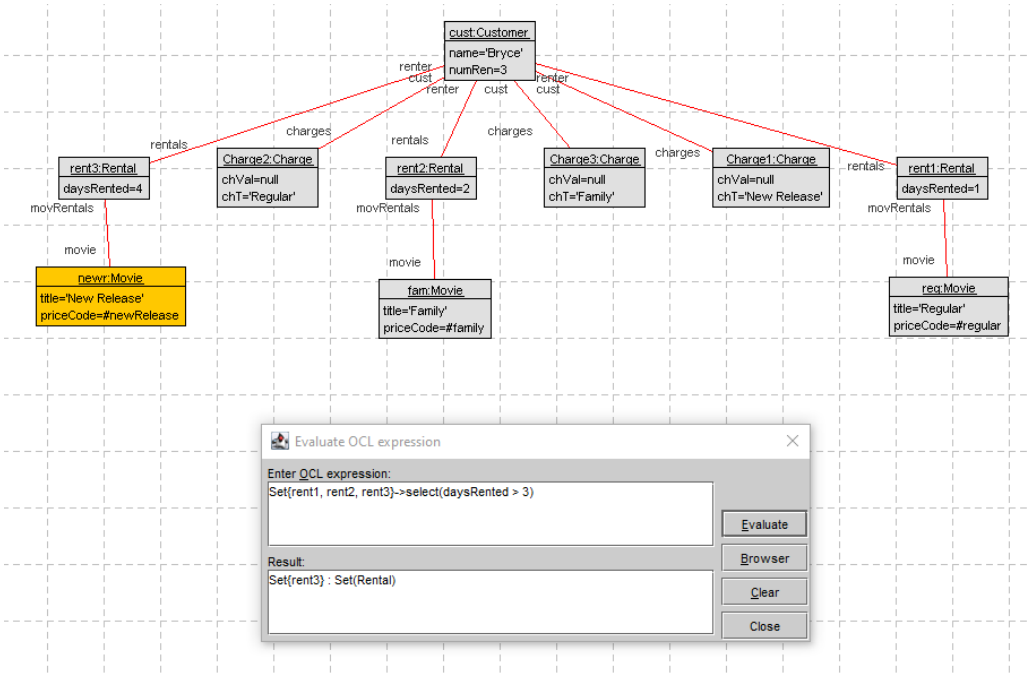
!set rent3.daysRented := 4

-- insert associations
!insert (cust, rent1) into custRentals
!insert (cust, rent2) into custRentals
!insert (cust, rent3) into custRentals

!insert (rent1, reg) into movRental
!insert (rent2, fam) into movRental
!insert (rent3, newr) into movRental

!cust.Statement()

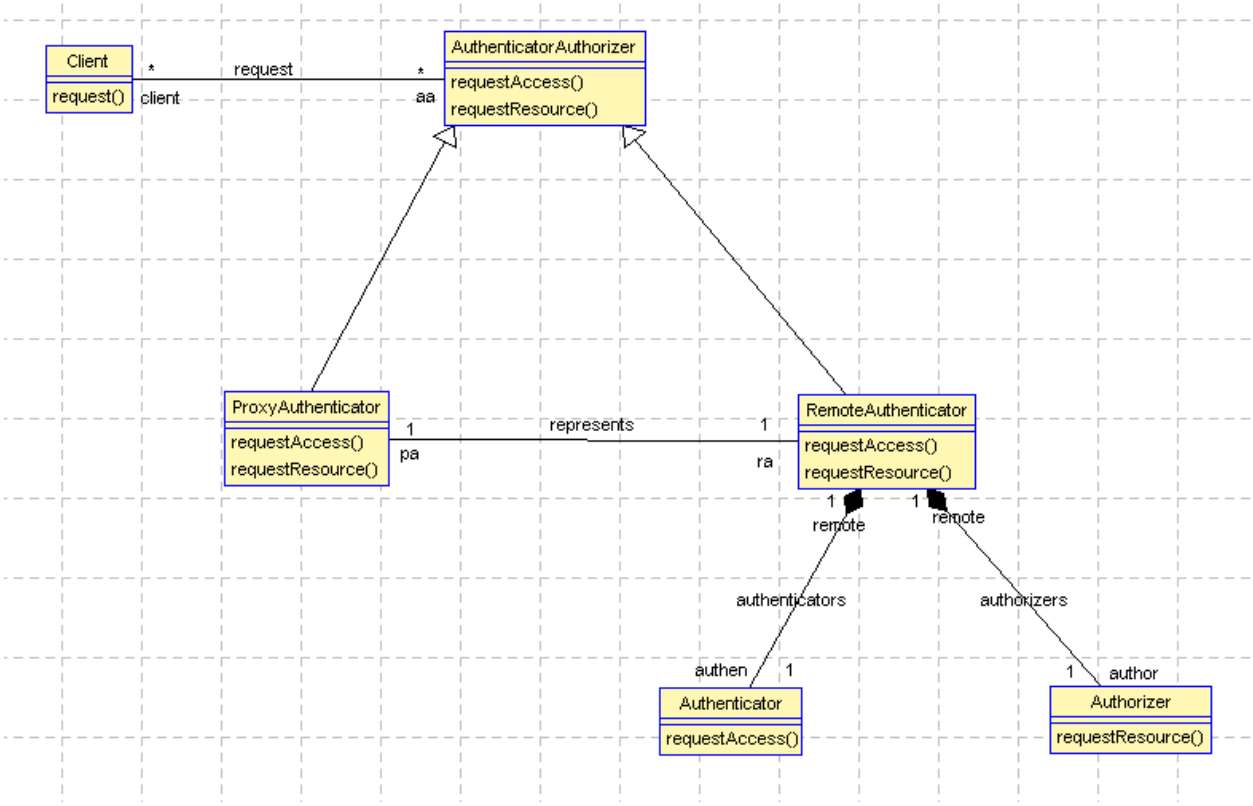


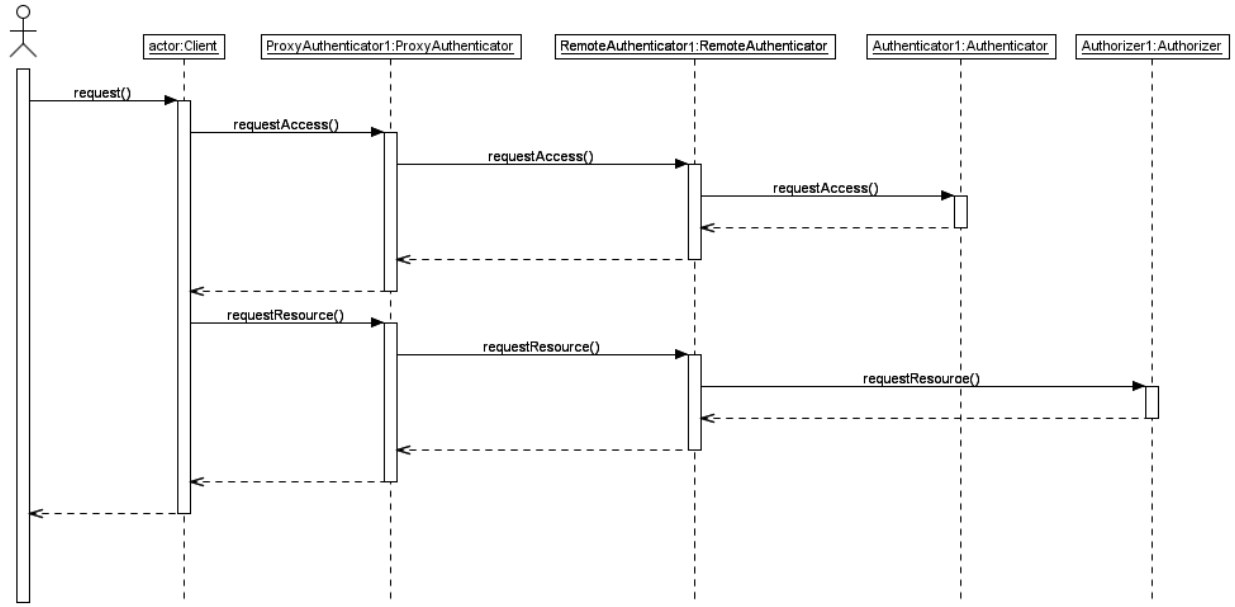



```
model RemoteAuthenticator

--classes

class Client
attributes
operations
   request()
      begin
         declare p:ProxyAuthenticator;
         p := new ProxyAuthenticator;
         p.requestAccess();
         p.requestResource();
      end
end

class AuthenticatorAuthorizer
attributes
operations
   requestAccess()
      begin
      end

   requestResource()
      begin
      end
end
```

```
class ProxyAuthenticator < AuthenticatorAuthorizer
attributes
operations
   requestAccess()
      begin
         declare r:RemoteAuthenticator;
         r := new RemoteAuthenticator;
         r.requestAccess();
      end

   requestResource()
      begin
         declare r:RemoteAuthenticator;
         r := new RemoteAuthenticator;
         r.requestResource();
      end
end

class RemoteAuthenticator < AuthenticatorAuthorizer
attributes
operations
   requestAccess()
      begin
         declare authen:Authenticator;
         authen := new Authenticator;
         authen.requestAccess();
      end

   requestResource()
      begin
         declare author:Authorizer;
         author := new Authorizer;
         author.requestResource();
      end
end

class Authenticator
operations
   requestAccess()
      begin
      end
end

class Authorizer
operations
   requestResource()
      begin
      end
end
```

```
--associations

association request between
   Client [*] role client
   AuthenticatorAuthorizer [*] role aa
end

association represents between
   ProxyAuthenticator [1] role pa
   RemoteAuthenticator [1] role ra
end

composition authenticators between
   RemoteAuthenticator [1] role remote
   Authenticator [1] role authen
end

composition authorizers between
   RemoteAuthenticator [1] role remote
   Authorizer [1] role author
end

-- initiate
!create actor:Client
!create proxy:ProxyAuthenticator
!create remote:RemoteAuthenticator
!create authenticator:Authenticator
!create authorizer:Authorizer

-- insert associations
!insert (actor, proxy) into request
!insert (actor, remote) into request
!insert (proxy, remote) into represents
!insert (remote, authenticator) into authenticators
!insert (remote, authorizer) into authorizers

!actor.request()
```


Client
request()
*
client
request
*
aa
AuthenticatorAuthorizer
requestAccess()
requestResource()
ProxyAuthenticator
requestAccess()
requestResource()
1
pa
represents
1
ra
RemoteAuthenticator
requestAccess()
requestResource()
1
remote
1
remote
authenticators
authorizers
authen
1
Authenticator
requestAccess()
1
author
Authorizer
requestResource()


actor:Client
ProxyAuthenticator1:ProxyAuthenticator
RemoteAuthenticator1:RemoteAuthenticator
Authenticator1:Authenticator
Authorizer1:Authorizer
request()
requestAccess()
requestAccess()
requestAccess()
requestResource()
requestResource()
requestResource()