Part 1

MovieRental.use file:

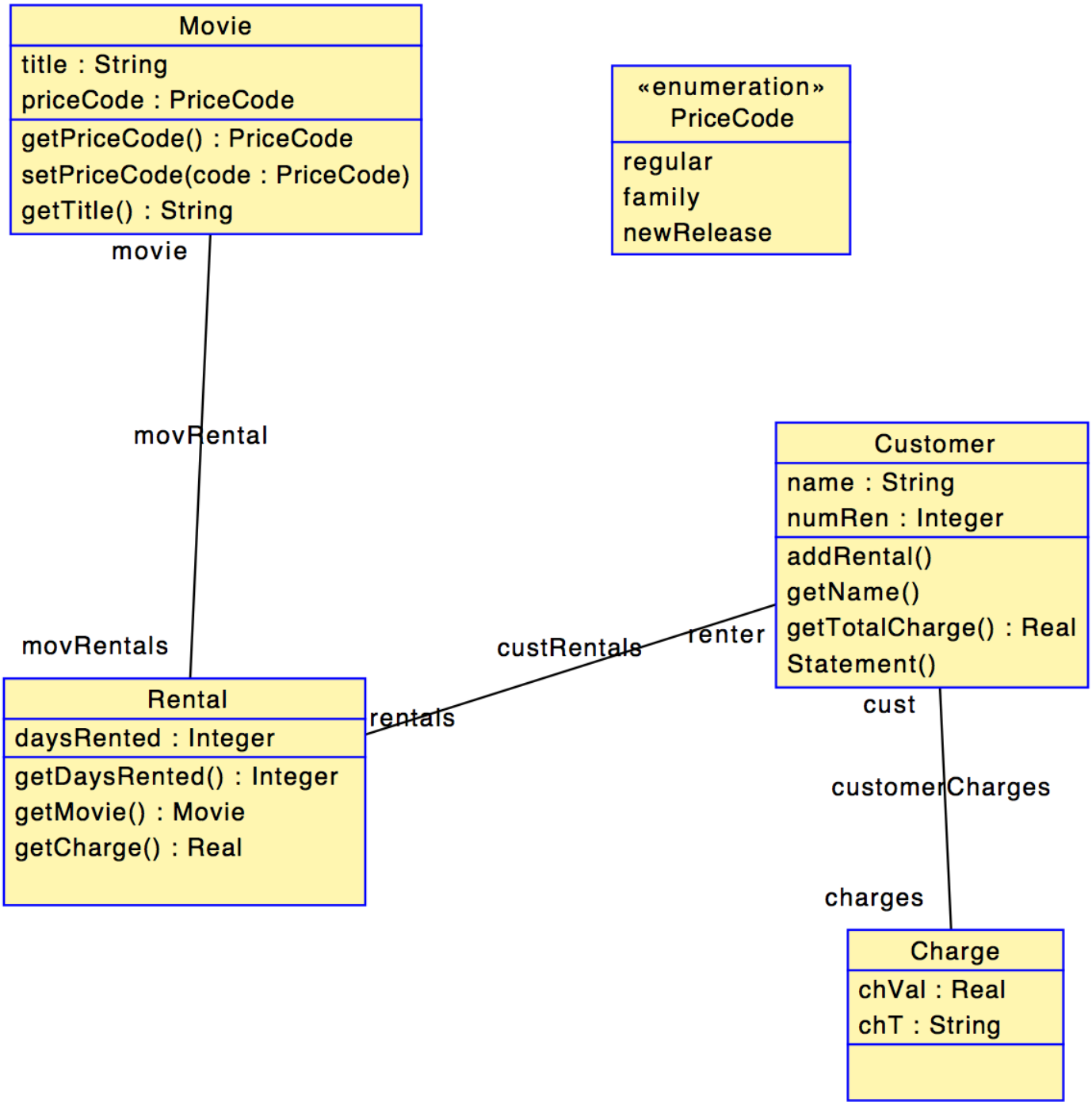
```
-- This is a USE model that has embedded SOIL operations in it
model MovieRental
enum PriceCode {regular, family, newRelease}
--classes
class Customer
attributes
    name:String
    numRen:Integer
operations
    addRental()
        begin
        end
    getName()
    getTotalCharge():Real
        begin
           declare sumCharge:Real, ch:Real;
           sumCharge := 0;
           for ren in self.rentals do
              ch := ren.getCharge();
              sumCharge := sumCharge + ch;
           end;
           result := sumCharge;
        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
```

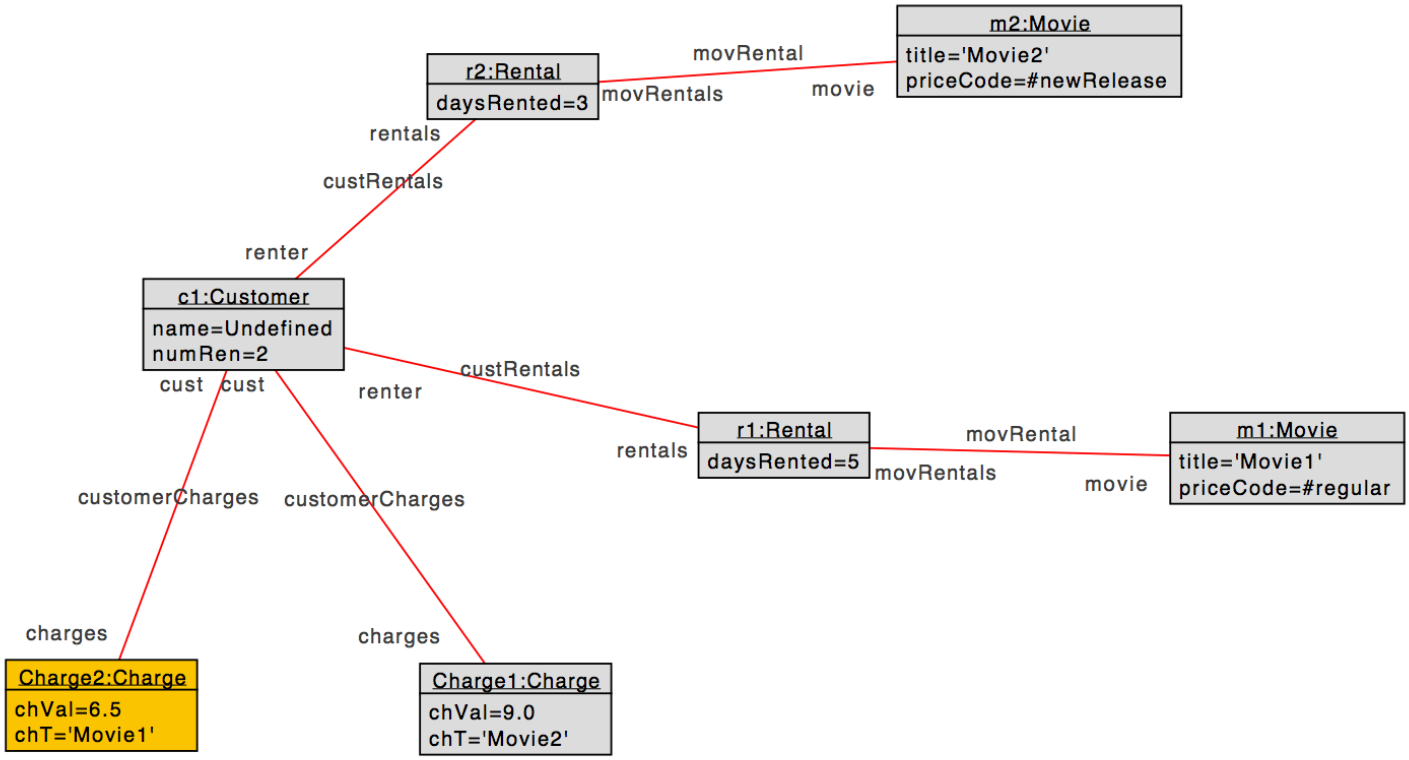
```
        begin
            result := self.daysRented;
        end
    getMovie(): Movie
        begin
            result := self.movie;
        end
    getCharge():Real
        begin
            declare wrkCh:Real, m:Movie, pc:PriceCode,dy:Integer;

            m:=self.getMovie();
            dy:=self.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
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
```

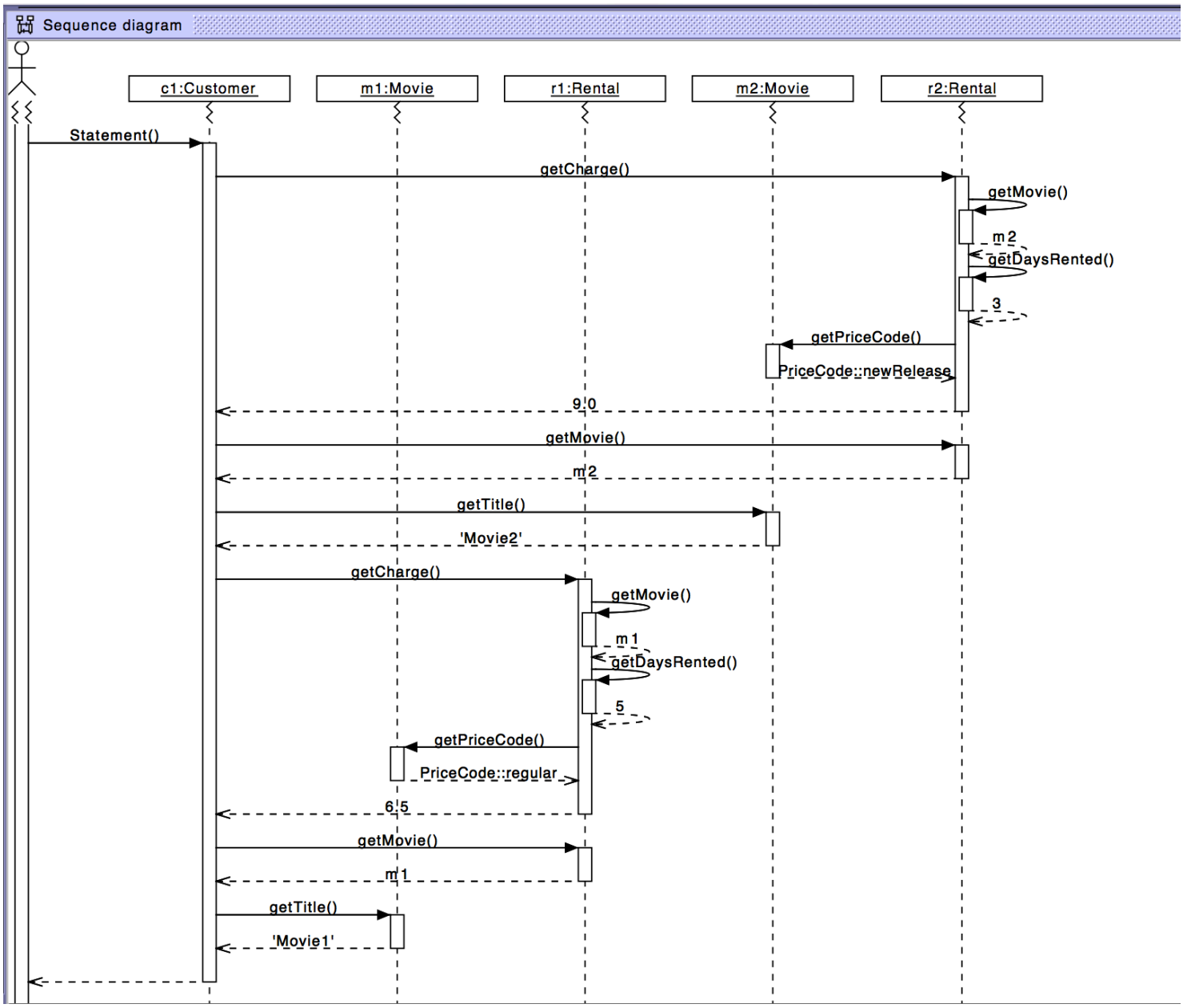
```
    chVal:Real
    chT: String
operations
end

--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
--Added for class exercises
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
```

MovieRental.x file:

```
!create c1:Customer
!create m1:Movie
!create r1:Rental
!set m1.priceCode := PriceCode::regular
!set m1.title := 'Movie1'
!set r1.daysRented := 5
!insert (c1,r1) into custRentals
!insert (r1,m1) into movRental
!create m2:Movie
!create r2:Rental
!set m2.priceCode := PriceCode::newRelease
!set m2.title := 'Movie2'
!set r2.daysRented := 3
!insert (c1,r2) into custRentals
!insert (r2,m2) into movRental
!c1.Statement()
```

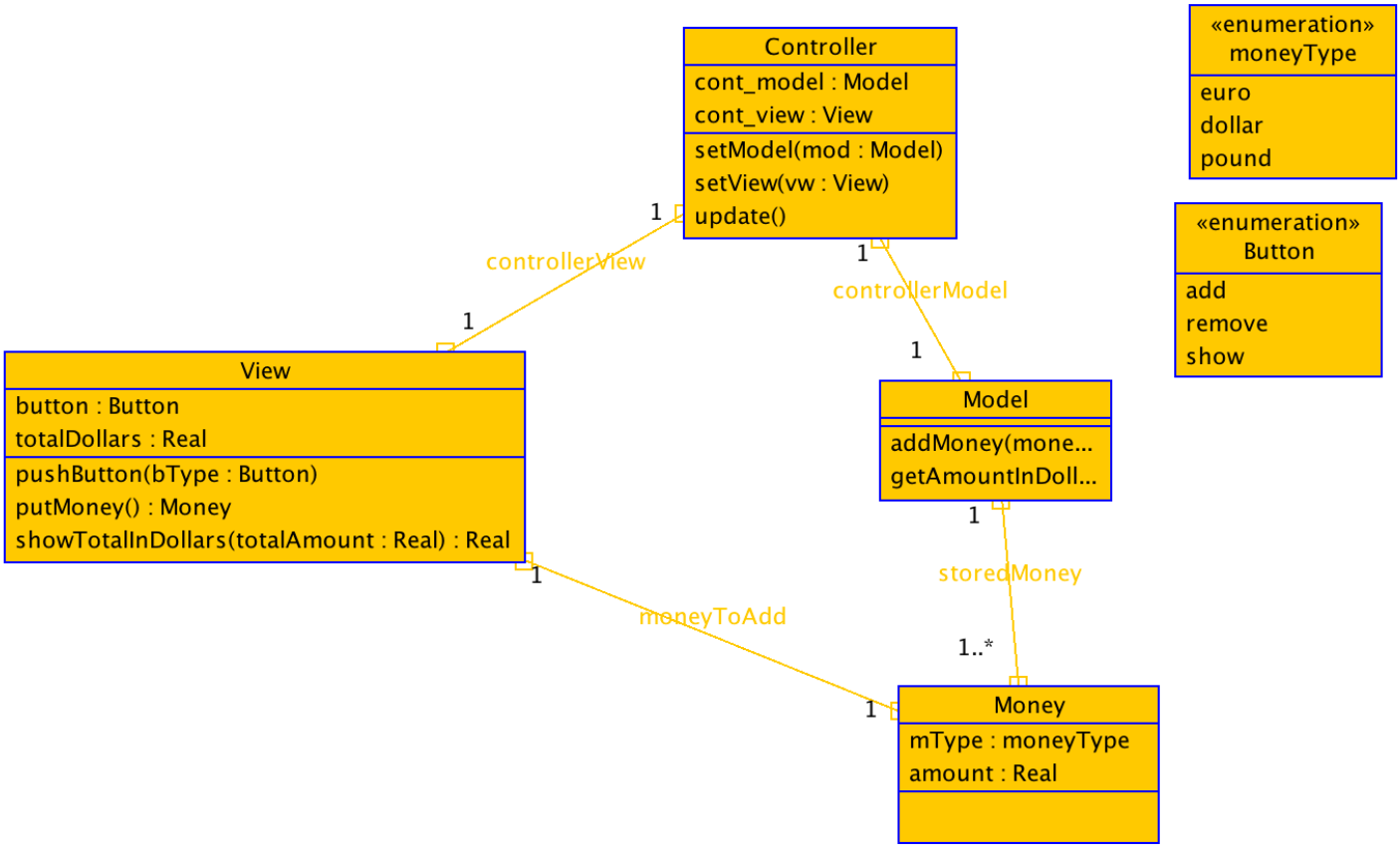

Movie
title : String
priceCode : PriceCode
getPriceCode() : PriceCode
setPriceCode(code : PriceCode)
getTitle() : String
«enumeration»
PriceCode
regular
family
newRelease
movie
movRental
movRentals
Customer
name : String
numRen : Integer
addRental()
getName()
getTotalCharge() : Real
Statement()
custRentals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

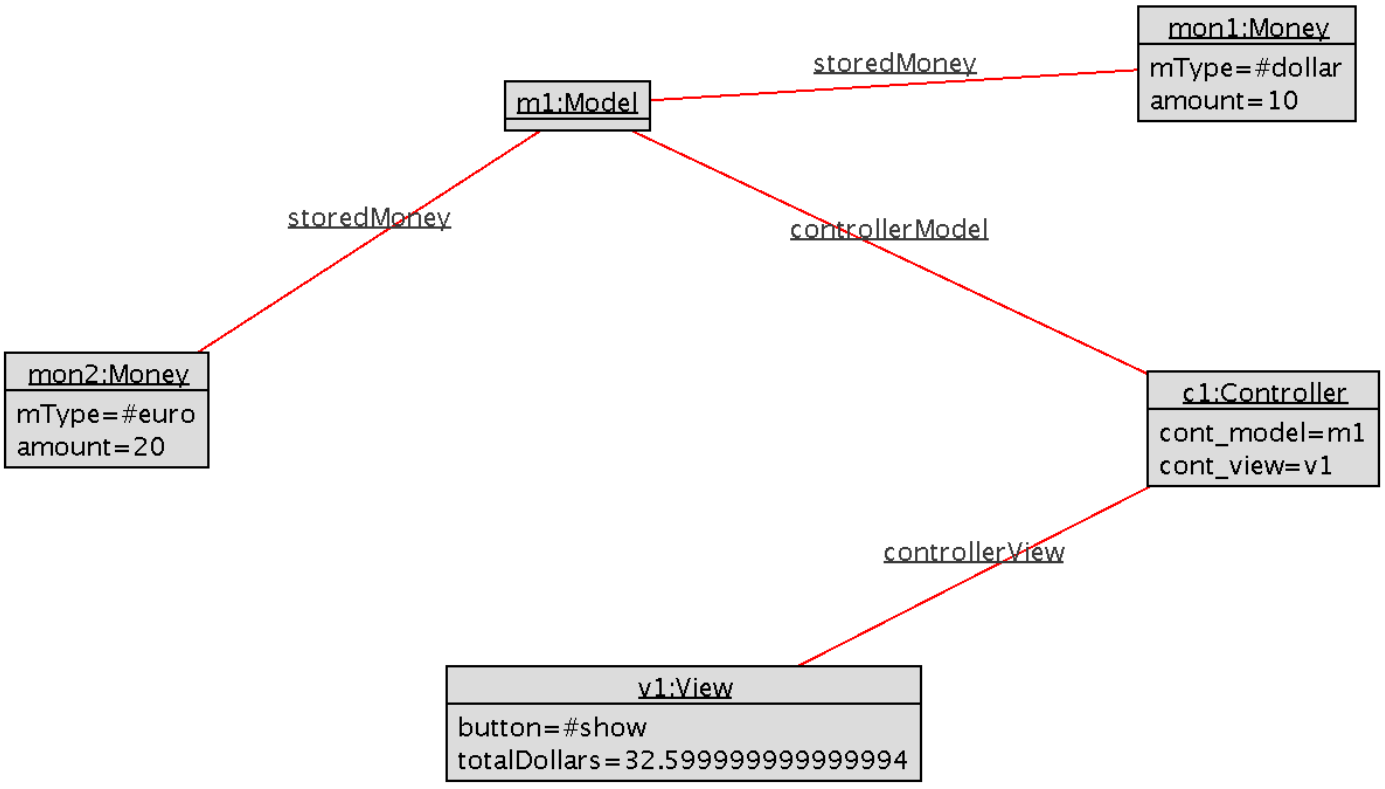

m2:Movie
title='Movie2'
priceCode=#newRelease
movRental
movie
r2:Rental
daysRented=3
movRentals
rentals
custRentals
renter
c1:Customer
name=Undefined
numRen=2
cust
cust
renter
custRentals
rentals
r1:Rental
daysRented=5
movRental
movRentals
movie
m1:Movie
title='Movie1'
priceCode=#regular
customerCharges
customerCharges
charges
charges
Charge2:Charge
chVal=6.5
chT='Movie1'
Charge1:Charge
chVal=9.0
chT='Movie2'

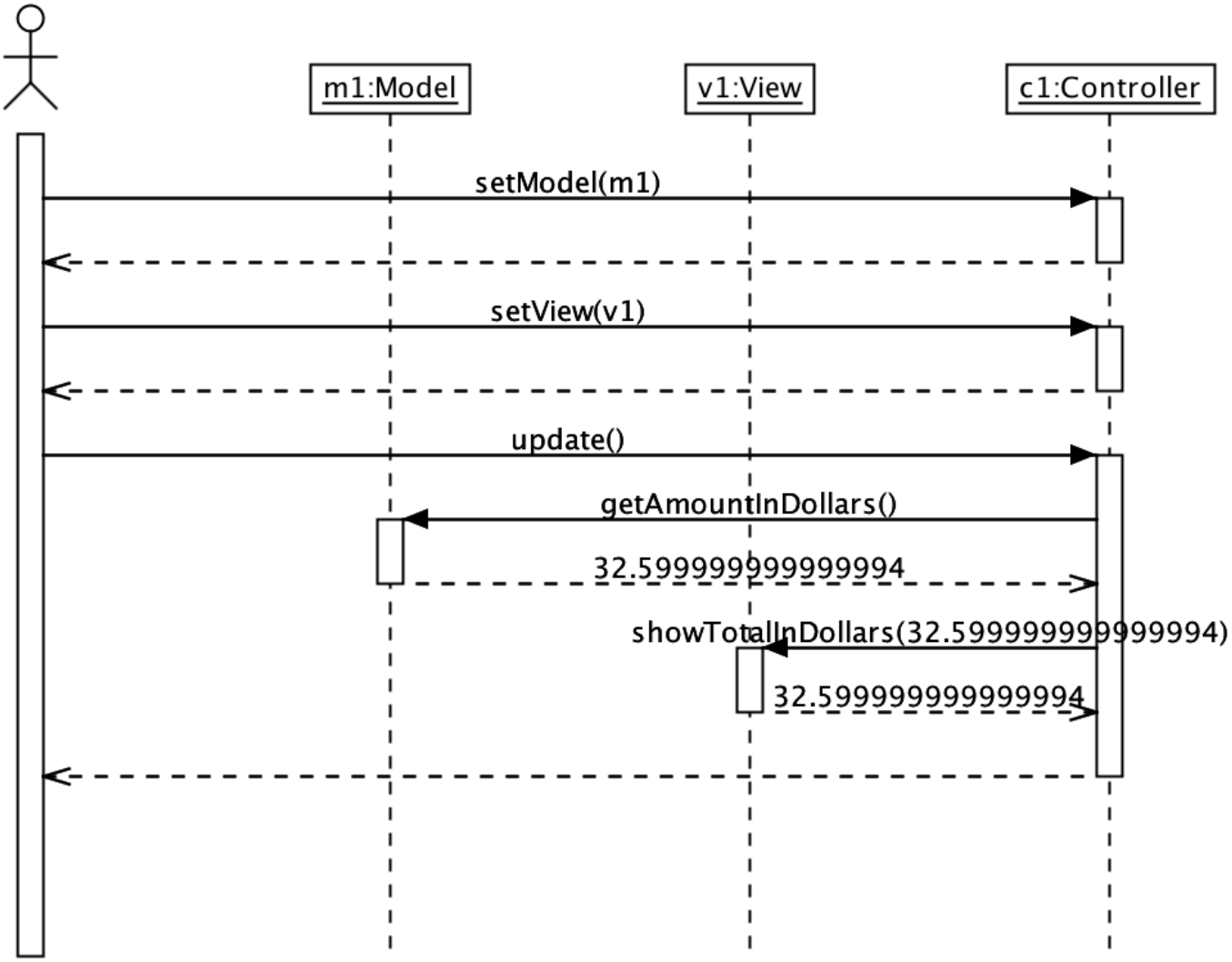


Part2:

We implemented the model view controller design pattern


Controller
cont_model : Model
cont_view : View
setModel(mod : Model)
setView(vw : View)
update()
«enumeration»
moneyType
euro
dollar
pound
«enumeration»
Button
add
remove
show
1
controllerView
1
1
controllerModel
1
View
button : Button
totalDollars : Real
pushButton(bType : Button)
putMoney() : Money
showTotalInDollars(totalAmount : Real) : Real
Model
addMoney(mone...
getAmountInDoll...
1
storedMoney
1
moneyToAdd
1..*
1
Money
mType : moneyType
amount : Real


mon1:Money
mType=#dollar
amount=10
storedMoney
m1:Model
storedMoney
controllerModel
mon2:Money
mType=#euro
amount=20
c1:Controller
cont_model=m1
cont_view=v1
controllerView
v1:View
button=#show
totalDollars=32.599999999999994



mvc.use file :

```
model MVC

enum Button {add, remove, show}
enum moneyType {euro, dollar, pound}

--classes

class Money
attributes
    mType:moneyType
    amount:Real
end
```

```
class Model
operations
    addMoney(moneyToAdd:Money)
        begin
        end
    getAmountInDollars():Real
        begin
            declare moneyTotal:Real;
            moneyTotal:=0;
            for money in self.mMonies do
                if money.mType=moneyType::euro then
                    moneyTotal := moneyTotal+money.amount*1.13;
                end;
                if money.mType=moneyType::dollar then
                    moneyTotal := moneyTotal+money.amount;
                end;
                if money.mType=moneyType::pound then
                    moneyTotal := moneyTotal+money.amount*1.45;
                end;
            end;
            result:=moneyTotal;
        end
end

class View
attributes
    button:Button
    totalDollars:Real

operations
    pushButton(bType:Button)
        begin
            self.button := bType;
        end
    putMoney():Money
        begin
            result:=self.vMonies;
        end
    showTotalInDollars(totalAmount:Real):Real
        begin
            self.totalDollars := totalAmount;
            result:=totalAmount;
        end
end
```

```
class Controller
attributes
    cont_model:Model
    cont_view:View
operations
    setModel(mod:Model)
        begin
            self.cont_model:=mod;
        end
    setView(vw:View)
        begin
            self.cont_view:=vw;
        end
    update()
        begin
            if self.view.button=Button::add then
                declare modelMoney:Money;
                modelMoney := self.view.putMoney();
                self.amodel.addMoney(modelMoney);
            end;
            if self.view.button=Button::show then
                declare totalDollars:Real;
                totalDollars:=self.amodel.getAmountInDollars();
                self.view.showTotalInDollars(totalDollars);
            end;
        end
end
```

```
--associations
association controllerView between
    Controller [1] role viewController
    View [1] role view
end
association controllerModel between
    Controller [1] role modelController
    Model [1] role amodel
end
association storedMoney between
    Model [1] role modelMoney
    Money [1..*] role mMonies
end
association moneyToAdd between
    View [1] role userMoney
    Money [1] role vMonies
end
--constraints
constraints
context Model
    inv maxMoney:mMonies->size <= 10
    inv minMoney:mMonies->size >= 1
    inv maxEuro:mMonies->select(mType = moneyType::euro)->size < 100.0
    inv maxPound:mMonies->select(mType = moneyType::pound)->size < 50.0
    inv maxDollar:mMonies->select(mType = moneyType::dollar)->size < 10.0
```

mvc.x file:

```
!create m1:Model
!create v1:View
!create c1:Controller
!create mon1:Money
!create mon2:Money
!insert (c1,v1) into controllerView
!insert (c1,m1) into controllerModel
!c1.setModel(m1)
!c1.setView(v1)
!set v1.button:=Button::show
!set mon1.mType:=moneyType::dollar
!set mon1.amount:=10
!set mon2.mType:=moneyType::euro
!set mon2.amount:=20
!insert (m1, mon2) into storedMoney
!insert (m1, mon1) into storedMoney
!c1.update()
```