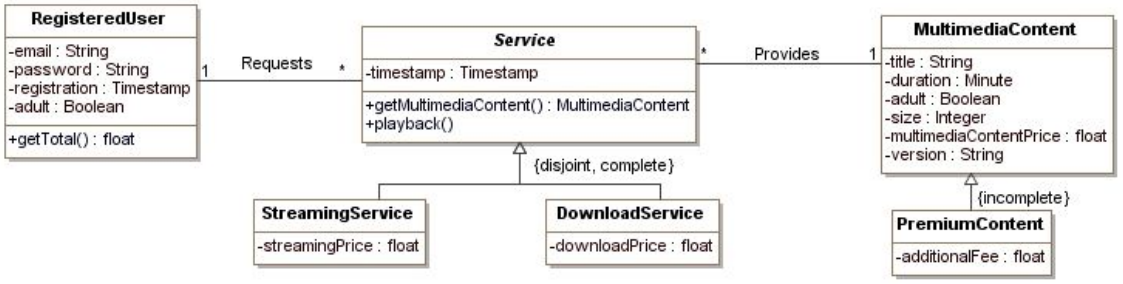
Anàlisi i Disseny amb Patrons

Pràctica 2: Principis d'assignació de responsabilitats i de disseny

Per poder elaborar aquesta Pràctica partirem del diagrama de classes de la Pràctica 1 amb algunes modificacions.

Recordeu que ens han contractat per desenvolupar un programari per gestionar el streaming de pel·lícules en línia. Com que a la solució publicada per la pràctica 1 hi havia diverses variants, mostrem aquí el disseny de partida, en el qual hem fet alguns canvis menors (comentats al peu dels diagrames).

El model del domini de partida és aquest:



Els canvis fets respecte al model del domini de la pràctica 1 són els següents:

- Hem canviat el preu dels serveis que demanen els usuaris. Els continguts multimèdia tindran un preu que es veurà incrementat amb l'*additionalFee* pel cas dels continguts premium. A més hi haurà un preu de streaming i un altre de download en funció del tipus de servei que els usuaris demanin. Això vol dir que diferents serveis poden tenir preus diferents en funció d'una sèrie de condicions que no tenim enregistrades al sistema (per exemple, si el servei és nocturn, si el client ha demanat molts serveis, etc…).
- Hem afegit l'operació *playback()* a la classe *Service* que inicia la reproducció/ descàrrega del contingut que proporciona el servei.
- Hem afegit la versió actual del contingut multimedia.

Exercici 1 (20%)

Suposem que l'empresa que s'encarrega de comercialitzar les pel·lícules en streaming (NetUOC) vol enregistrar en el sistema el % de benefici que cobrarà a tot servei que els usuaris registrats demanin. Es demana:

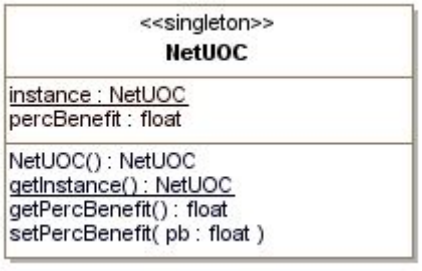
a) Quin patró de disseny és adequat per enregistrar aquest % de benefici?
b) Fes el diagrama de classes resultant d'aplicar aquest patró (només cal la part d'aplicació del patró)
c) Fes el pseudocodi del diagrama de classes que proposes a l'apartat anterior.

Nota:

- El % de benefici serà utilitzat més endavant en la pràctica per calcular el preu de cada servei. No cal que en aquest exercici tingueu en compte aquest càlcul.

Solució:

a) Aplicarem el patró *Singleton* a la classe *NetUOC* per tal de mantenir una única instància d'aquesta classe que tingui definit el % de benefici que vol obtenir l'empresa per a cada servei que sigui proveït. Aquest % de benefici s'emmagatzemarà una única vegada per tots els serveis.
b) El diagrama de classes resultant del patró tindrà definida una única classe.



c)

```
public class NetUOC {
        private static NetUOC instance = null;
        private float percBenefit;
        private NetUOC() {}
```

```
        public static NetUOC getInstance() {
                if (instance == null) {
                        instance = new NetUOC();
                }
                return NetUOC;
        }
        public float getPercBenefit() {
                return percBenefit;
        }
        public float setPercBenefit(pb) {
                percBenefit = pb;
        }
}
```

Exercici 2 (20%)

Suposem que NetUOC vol calcular el preu de cada servei amb l'operació *getPrice(): Float* definida a a la classe *Service*. Si el servei és un servei de streaming i proporciona un contingut multimedia no premium llavors el preu serà: streamingPrice + multimediaContentPrice + %benefici (de NetUOC). Si el servei és un servei de streaming i proporciona un contingut premium llavors el preu serà: streamingPrice + multimediaContentPrice + additionalFee + %benefici (de NetUOC). Si el servei és un servei de download i proporciona un contingut multimedia no premium llavors el preu serà: downloadPrice + multimediaContentPrice + %benefici (de NetUOC). Si el servei és un servei de download i proporciona un contingut premium content llavors el preu serà: downloadPrice + multimediaContentPrice + additionalFee + %benefici (de NetUOC). Es demana:

a) Quin patró de disseny és adequat aplicar en aquesta situació?
b) Fes el diagrama de classes amb les operacions resultants d'aplicar aquest patró (podeu proporcionar només la part d'aplicació del patró)

c) Fes el pseudocodi o el diagrama de seqüència de l'operació *getPrice():Float* i de totes les operacions que siguin invocades des d'aquesta operació.

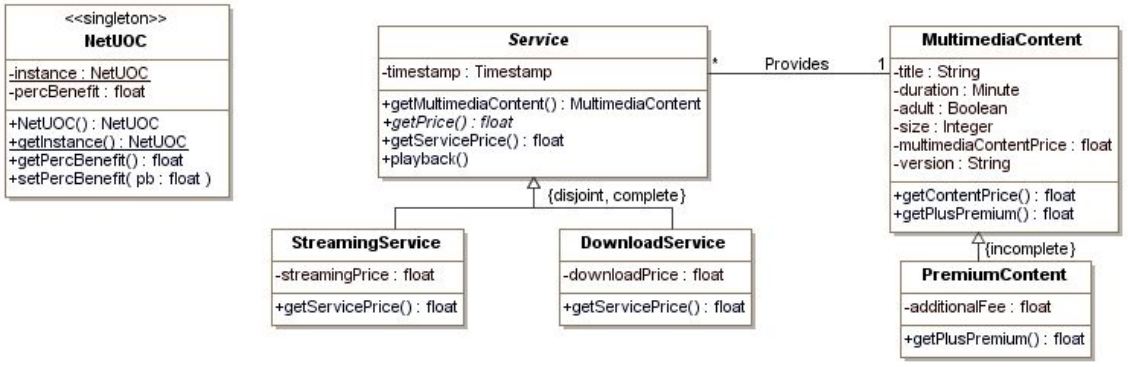
Nota:

- Es valorarà especialment que a la solució proposada no hi hagi repetició de codi.

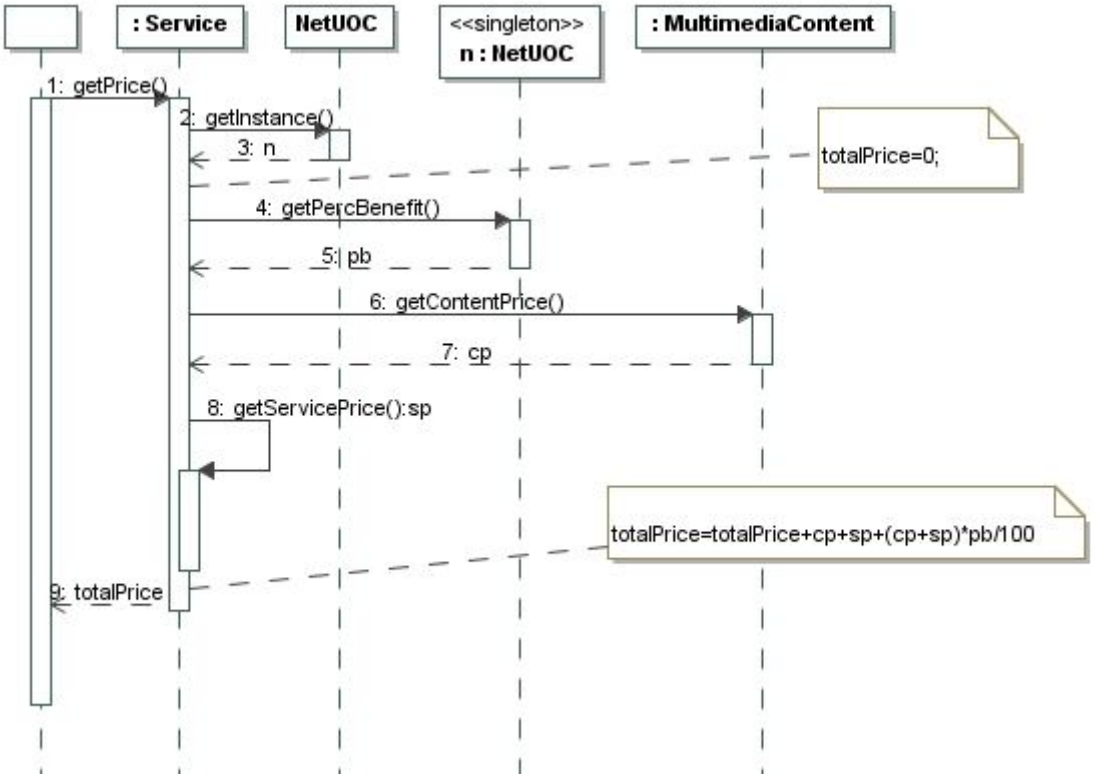
Solució:

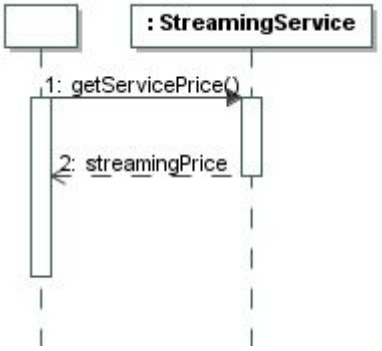
a) Aplicarem el patró *mètode Plantilla* dos cops, al mètode que calcula el preu d'un MultimediaContent i al mètode *getPrice()* que calcularà el preu d'un servei. Al primer mètode la part comuna correspon a l'obtenció del multimediaContentPrice i la part específica a l'obtenció de l'additionalFee quan el MultimediaContent és un PremiumContent. Al segon mètode la part comuna correspon al càlcul del preu del multimedia content més el % benefici de NetUOC i la part específica correspon a l'obtenció del streamingPrice o downloadPrice (en funció del tipus de servei). Com es pot comprovar a l'apartat c), aquesta solució evita la repetició de codi ja que la part comuna de les operacions es defineix un únic cop.

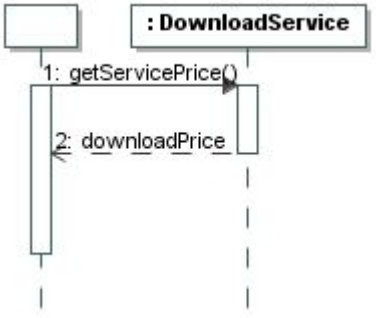
b) El diagrama de classes resultant de l'aplicació del patró mètode plantilla a les operacions *getPrice()* i *getContentPrice()* es pot trobar a continuació.

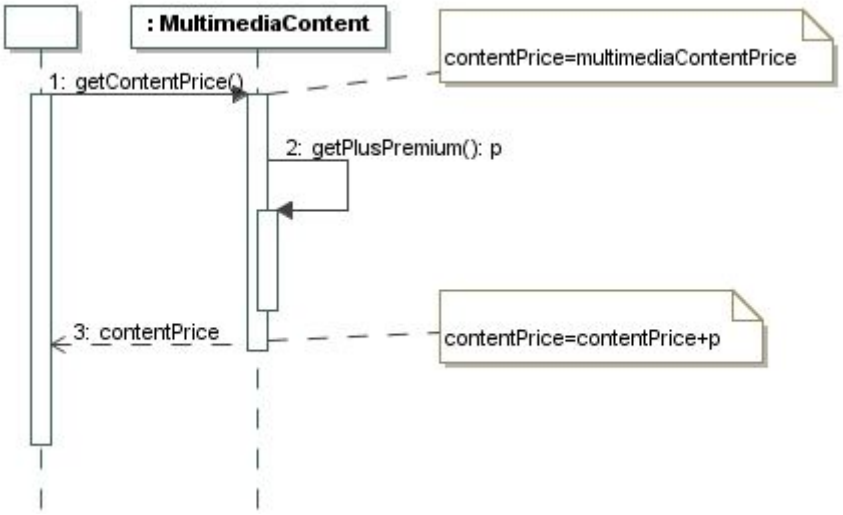


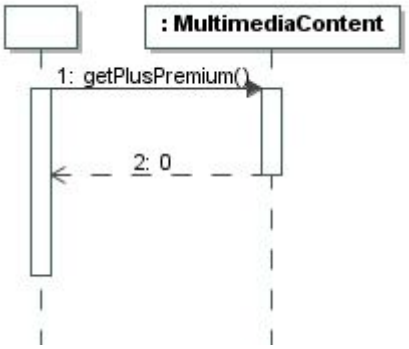
c)

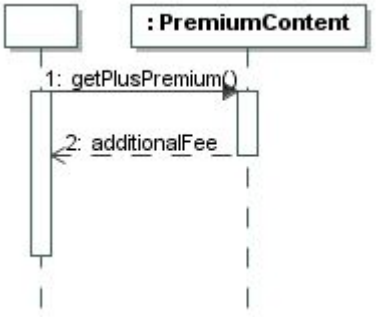

: Service
NetUOC
<<singleton>>
n : NetUOC
: MultimediaContent
1: getPrice()
2: getInstance()
3: n
totalPrice=0;
4: getPercBenefit()
5: pb
6: getContentPrice()
7: cp
8: getServicePrice():sp
totalPrice=totalPrice+cp+sp+(cp+sp)*pb/100
9: totalPrice


: StreamingService
1: getServicePrice()
2: streamingPrice


: DownloadService
1: getServicePrice()
2: downloadPrice







Exercici 3 (20%)

Suposem que volem dissenyar l'operació createDownloadServiceWithPremiumContent(ts: Timestamp, dlPrice: float, pc:PremiumContent): Integer de la classe *RegisteredUser*. Aquesta operació que crea un servei de download per a l'usuari registrat amb un contingut premium té les següents precondicions (condicions que es satisfan abans d'invocar l'operació):

- No existeix un objecte *DownloadService* amb les dades dels paràmetres de l'operació.
- El paràmetre *downloadPrice* és més gran que 0.

L'operació ha de fer el següent:

- Crear un objecte *DownloadService ds* i assignar-li el *downloadPrice dlPrice* i el *timestamp ts* i el *PremiumContent pc*.
- Guardar al registered user l'objecte *ds* creat.
- Calcular i retornar el nombre total de serveis de download amb contingut premium que té l'usuari registrat.

Es demana:

a) A quina classe correspon la responsabilitat d'implementar el càlcul el nombre total de serveis de download amb contingut premium que té l'usuari registrat? Justifica la resposta.
b) Quins patrons de disseny és adequat aplicar en aquesta situació?
c) Fes el pseudocodi o el diagrama de seqüència de l'operació *createDownloadServiceWithPremiumContent* i de totes les operacions que siguin invocades des d'aquesta operació.

Nota:

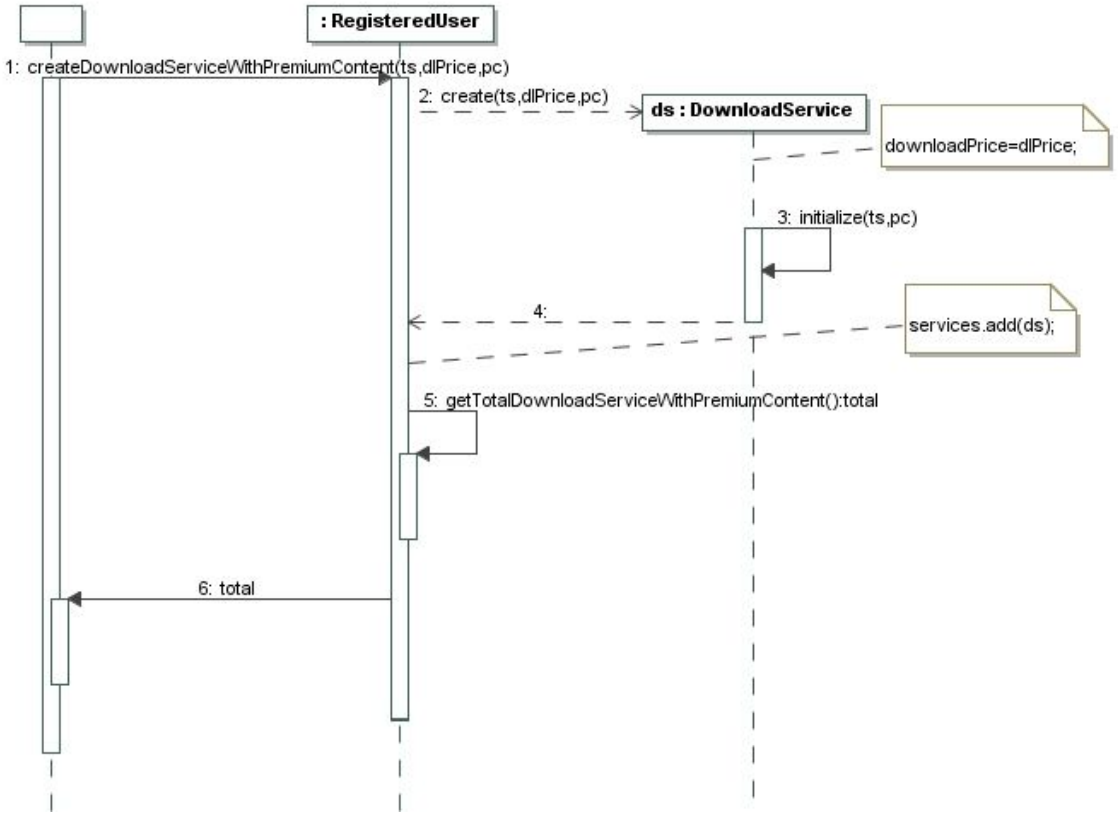
- Ens diuen que el càlcul del nombre total de serveis de download amb contingut premium que té un usuari registrat també és necessari per altres operacions del sistema que estem desenvolupant. Es valorarà especialment que la vostra solució sigui reusable.

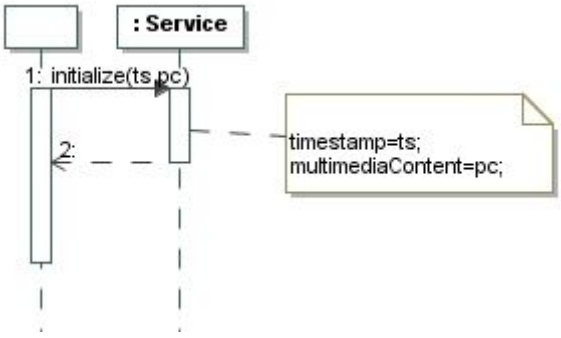
Solució:

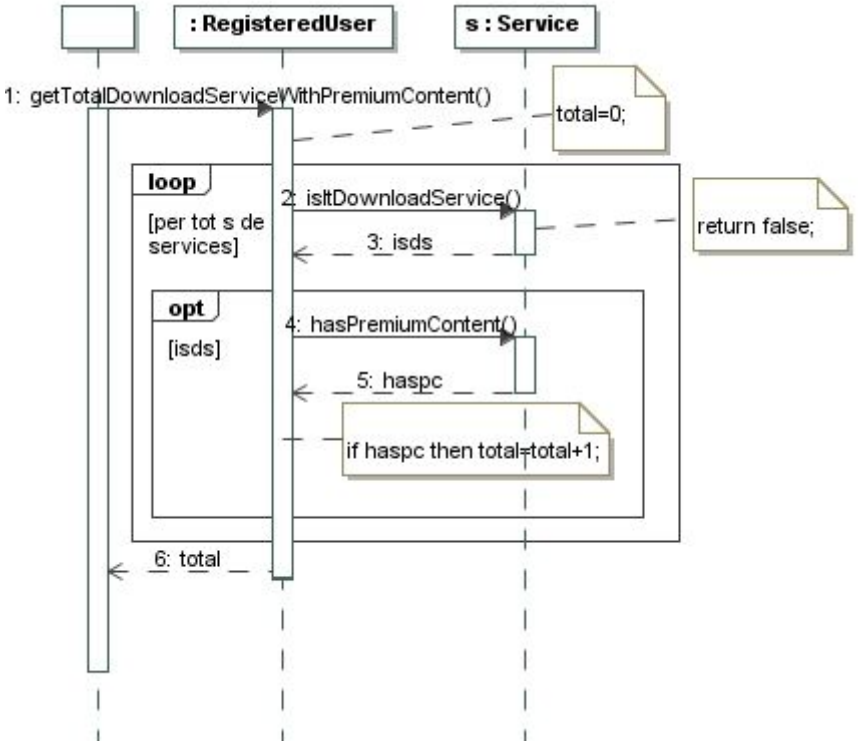
a) La classe que conté la informació necessària per poder fer aquest càlcul és la classe *RegisteredUser*. Aquesta classe té definits els serveis que l'usuari registrat ha demanat i, per tant, aplicant el patró *Expert* és la classe on s'ha de definir aquest càlcul.

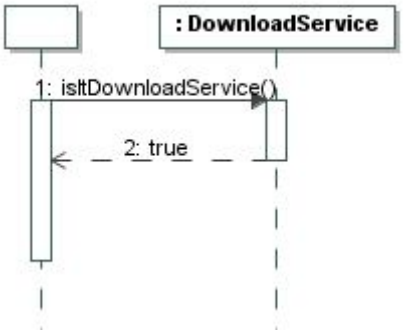
b) Hem d'aplicar el patró *Creador* i assignar a la classe *RegisteredUser* la responsabilitat de crear un servei de download ja que aquesta classe conté els objectes serveis.

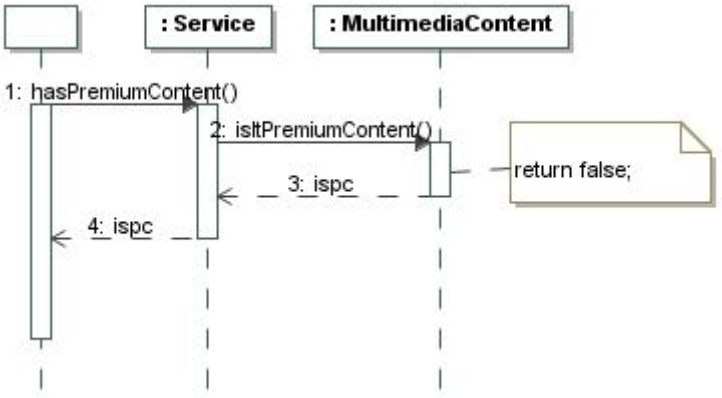
c) A continuació es mostra el diagrama de seqüencia de l'operació *createDownloadServiceWithPremiumContent.* Noteu que s'ha creat una operació per fer el càlcul del nombre total de serveis de download amb contingut premium que té un usuari registrat per tal de que aquesta operació pugui ser reusada en altres operacions del sistema.

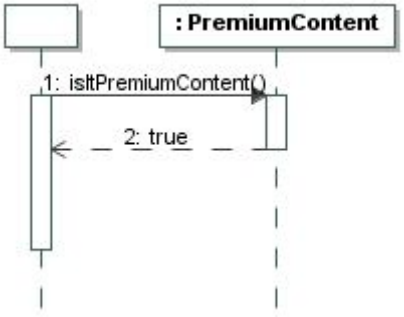



: Service
1: initialize(ts,pc)
2:
timestamp=ts;
multimediaContent=pc;


: RegisteredUser
s : Service
1: getTotalDownloadServiceWithPremiumContent()
total=0;
loop
[per tot s de services]
2: isItDownloadService()
return false;
3: isds
opt
[isds]
4: hasPremiumContent()
5: haspc
if haspc then total=total+1;
6: total


: DownloadService
1: isItDownloadService()
2: true





Exercici 4 (20%)

NetUOC ens demana fer una modificació en els requisits del sistema que estem desenvolupant. Fins ara els usuaris registrats podien reproduir/descarregar les vegades que volguessin els continguts associats als seus serveis utilitzant l'operació *playback()* de la classe *Service*. A partir d'ara es vol que determinats serveis tinguin un límit de 4 reproduccions/descàrregues dels seus continguts i d'altres es puguin reproduir/descarregar tantes vegades com l'usuari desitji (la decisió de si hi ha un límit de reproduccions es pren en el moment de la creació del servei i la seva assignació al usuari registrat, i queda fora de l'abast d'aquest exercici). Heu de considerar que no podeu modificar l'operació *playback()* de la classe *Service*. Es demana:

a) Quin patró de disseny és adequat aplicar en aquesta situació?
b) Fes el diagrama de classes amb les operacions resultants d'aplicar aquest patró (podeu proporcionar només la part d'aplicació del patró)

c) Fes el pseudocodi o el diagrama de seqüència de l'operació *playbackAllContents()* de la classe *RegisteredUser* que reprodueix/descarrega tots els continguts associats als seus serveis i de totes les operacions que siguin invocades des d'aquesta operació.

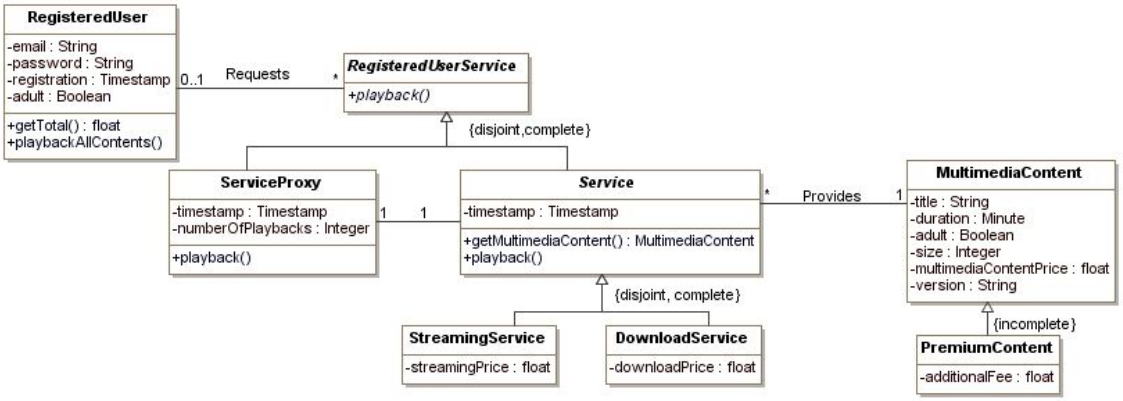
Nota:

- Si un usuari registrat intenta reproduir/descarregar un contingut que té un límit de reproduccions/descàrregues i ja ha assolit el límit, no es farà res. En cas contrari, es reproduirà/descarregarà el contingut mitjançant la invocació de l'operació *playback()*.

Solució:

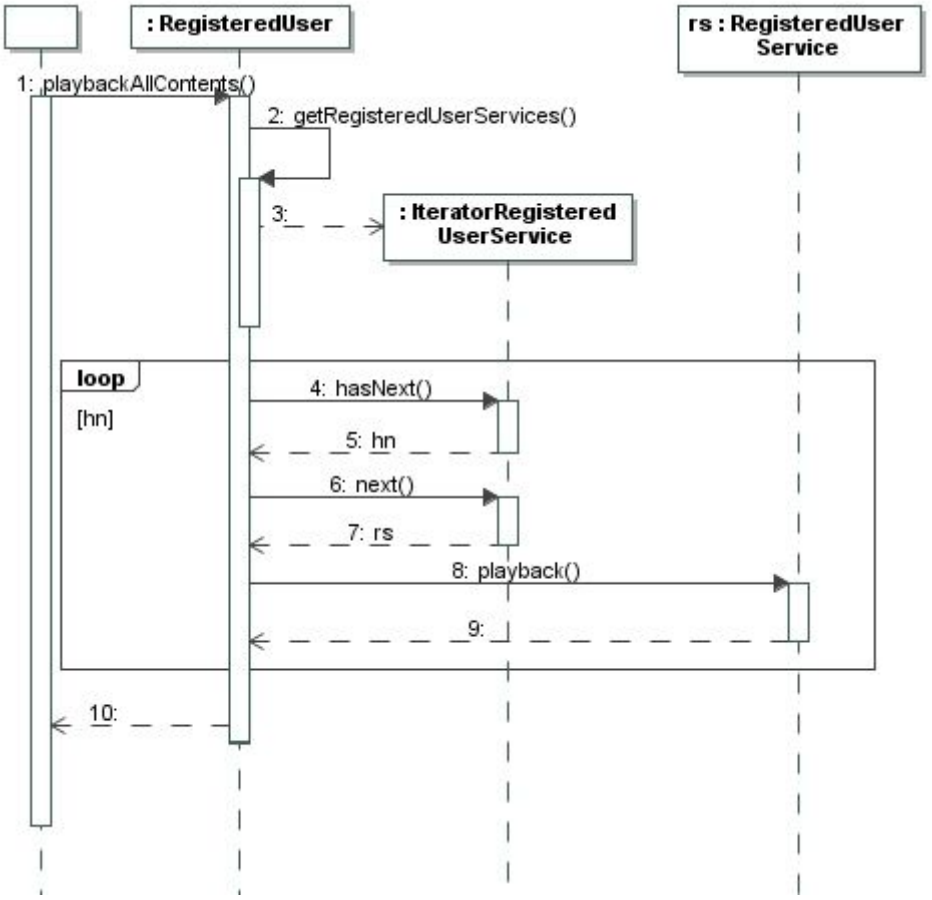
a) El patró principal que usarem per tractar aquesta situació és el patró *Representant*. Per als serveis que tinguin un límit de descàrregues, el representant controlarà si s'ha arribat al límit de reproducció/descàrrega (invocació de l'operació *playback()*). Per a la resta de serveis, la reproducció/descàrrega es farà invocant directament a l'operació *playback()*.

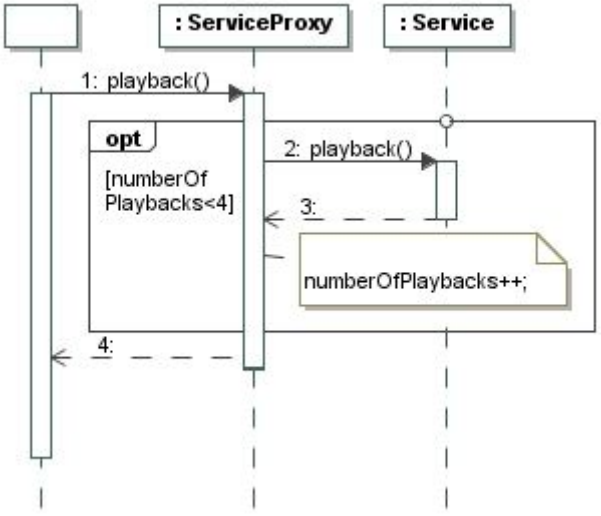
b) A continuació disposeu de la part del diagrama de classes que hem modificat per aplicar el patró *Representant*.



Nota: Una altra solució podria ser deixar l'atribut timestamp a la classe *RegisteredUser* i no definir-lo a les subclasses.

c) A continuació es mostra el diagrama de seqüencia de l'operació *playbackAllContents()*.


: RegisteredUser
rs : RegisteredUser Service
1: playbackAllContents()
2: getRegisteredUserServices()
3:
: IteratorRegistered UserService
loop
[hn]
4: hasNext()
5: hn
6: next()
7: rs
8: playback()
9:
10:


: ServiceProxy
: Service
1: playback()
opt
[numberOf Playbacks<4]
2: playback()
3:
numberOfPlaybacks++;
4:

Exercici 5 (20%)

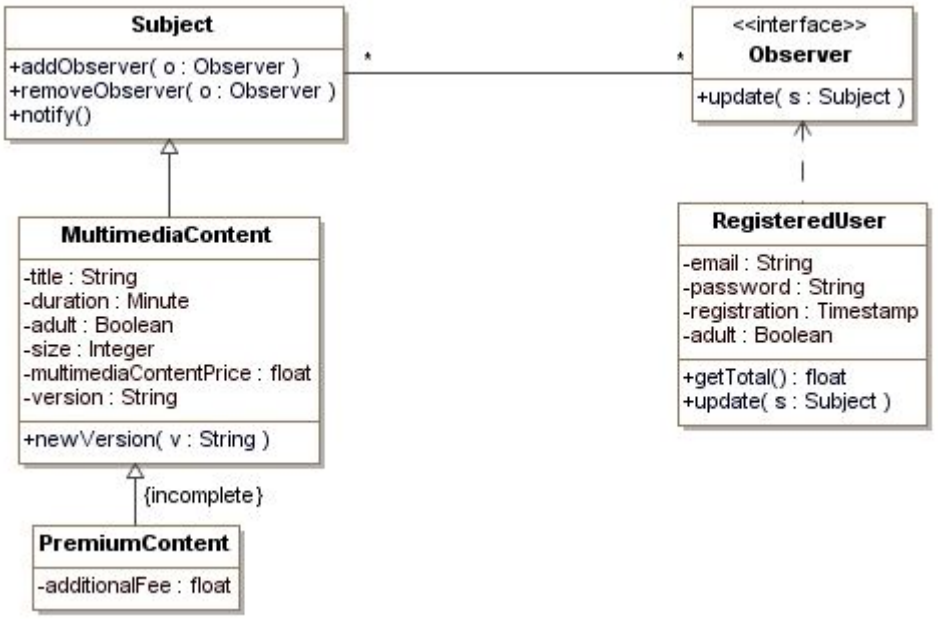
Suposem que NetUOC vol oferir als seus usuaris registrats la possibilitat de subscriure's a diferents continguts per tal de ser notificats via mail quan arribi una nova versió del mateix. Cada vegada que hi hagi una nova versió d'un contingut, els usuaris registrats que estiguin subscrits rebran un mail amb el text següent: "The new version "+ version + " of the multimedia content " + title + " is available.". Podeu suposar l'existència de l'operació sendMail(email: String, text: String) que pot ser invocada des d'on vulgueu i que envia el mail amb el text indicat al paràmetre a l'adreça email indicada al paràmetre. Es demana:

a) Quin patró de disseny és adequat aplicar en aquesta situació?
b) Fes el diagrama de classes amb les operacions resultants d'aplicar aquest patró (podeu proporcionar només la part d'aplicació del patró)
c) Fes el pseudocodi o el diagrama de seqüència de l'operació *newVersion(v: String)* de la classe *MultimediaContent* que actualitza la nova versió *v* del contingut i notifica als seus subscrits aquesta actualització. Cal que feu també el diagrama de seqüència de totes les operacions que siguin invocades des d'aquesta operació.

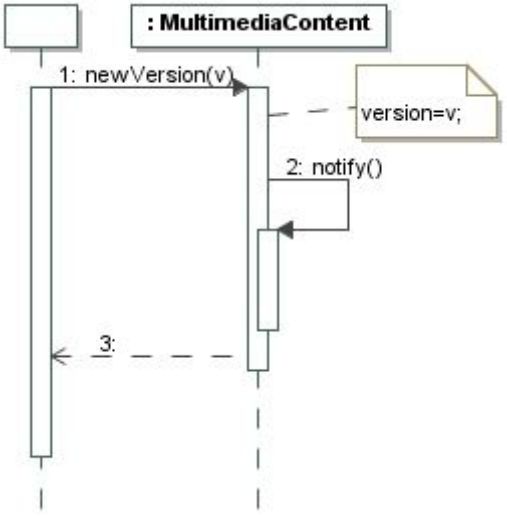
Nota: No volem que la solució acobli els continguts multimèdia amb els usuaris registrats.

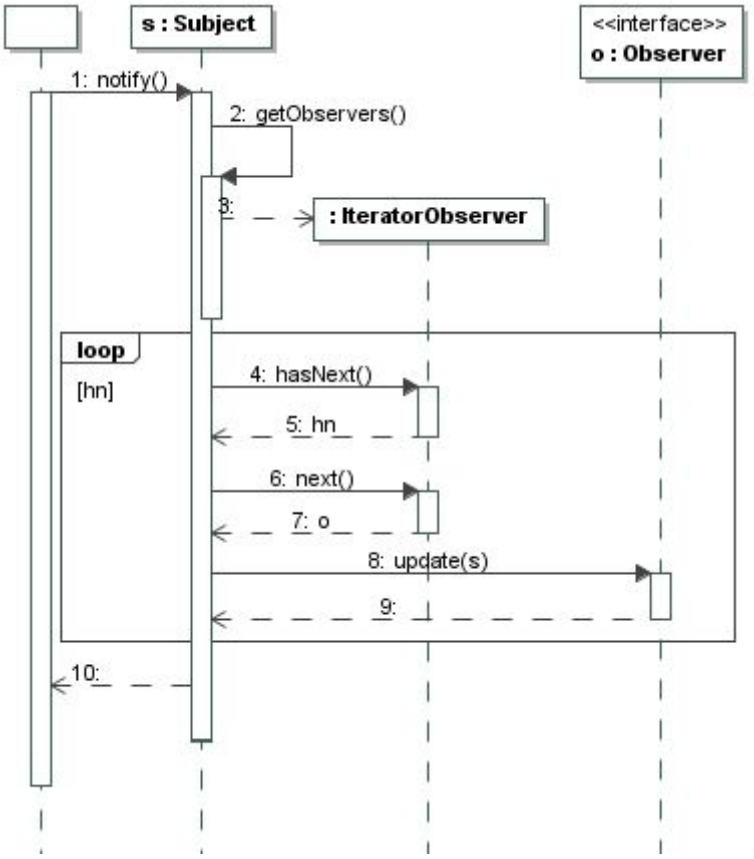
Solució:

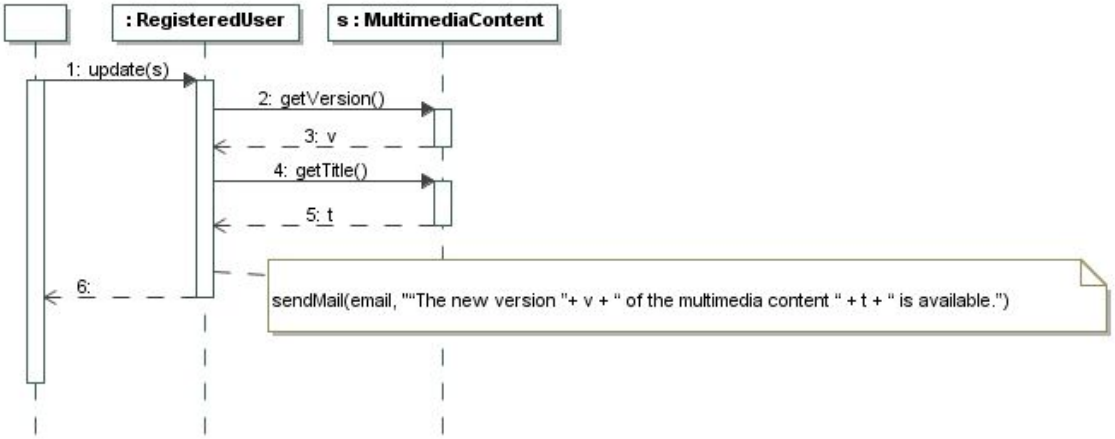
a) El patró principal que usarem per tractar aquesta situació és el patró *Observador*. Quan arribi una nova versió d'un contingut l'operació notificar del patró s'encarregarà d'actualitzar tots els observadors (*RegisteredUser*) que estiguin associats amb el contingut. L'operació d'actualitzar serà l'encarregada d'enviar el mail.
b) A continuació disposeu de la part del diagrama de classes on hem aplicat el patró *Observador*.



c) A continuació es mostra el diagrama de seqüencia de l'operació *newVersion(v: String)*.




s : Subject
<<interface>>
o : Observer
1: notify()
2: getObservers()
3:
: IteratorObserver
loop
[hn]
4: hasNext()
5: hn
6: next()
7: o
8: update(s)
9:
10:


: RegisteredUser
s : MultimediaContent
1: update(s)
2: getVersion()
3: v
4: getTitle()
5: t
6:
sendMail(email, ""The new version "+ v + " of the multimedia content " + t + " is available.")