Classes d'anàlisi

Durant el procés de desenvolupament del software, el principal objectiu de la fase d'anàlisi és l'aprofundiment en la comprensió dels requeriments capturats pel model de casos d'ús. En aquesta fase s'acostuma a fer servir més el punt de vista informàtic que no pas el del usuari, de manera que és possible modelar el sistema més formalment.

Les classes d'anàlisi representen un model inicial del sistema. A la pràctica una classe d'anàlisi acaba dividint-se en vàries classes de disseny. Una classe d'anàlisi sempre s'ajusta a un dels tres estereotips bàsics: **entitat, frontera** i **control**.

Exemple

El següent cas d'ús representa la funcionalitat que permet a un empleat de banca transferir diners des d'un compte a un altre:



El procés consta de dos passos:

1. Treure diners del **Compte1**;
2. Dipositar diners al **Compte2**.

Els objectes **Compte1** i **Compte2** de la classe **Compte** no saben res un de l'altre. Per tant el primer compte normalment no tindrà cap relació amb el segon. A més a més probablement voldrem enregistrar la transferència. Podríem fer que un dels comptes fos el responsable d'enregistrar la transferència, però això tampoc seria natural. Per a solucionar-ho podem introduir un nou objecte **Transferir** que serà el responsable de la transferència. L'empleat demanarà a l'objecte **Transferir** que faci la transferència i aquest traurà diners d'un compte i els dipositarà a l'altre. L'objecte **Transferir**, afegit per l'analista, pren la responsabilitat de la transacció. A més a més pot ocupar-se també d'enregistrar-la. D'aquesta forma amb els comptes es fa allò que se suposa que s'ha de fer, treure-hi i dipositar-hi diners. Normalment, a la fase d'anàlisi, un cas d'ús tindrà un objecte de control d'aquest tipus. Amb la solució anterior encara hi ha un problema: a la realitat l'empleat no pot interactuar directament amb l'objecte **Transferir**. Haurem de dissenyar una interfície de pantalla, l'objecte **PantallaTransferir**, de manera que l'empleat hi pugui interactuar entrant-hi els detalls necessaris per a poder fer la transferència. Després **PantallaTransferir** demanarà a **Transferir** que faci la transferència.

Diferents tipus de classes d'anàlisi

Classes d'Entitats

Les classes d'entitats contenen els objectes bàsics que corresponen al mon real. Normalment s'emmagatzemen a una base de dades. Sovint són independents de l'aplicació. Generalment els objectes d'una d'aquestes classes no necessiten conèixer res uns del altres. Es comuniquen a través dels objectes de les classes de control. Els objectes de las classes d'entitats només tenen coneixement de les coses que poden fer per a ells mateixos.

Classes de Control

Cada cas d'ús ha de tenir almenys un objecte d'una classe de control que s'ocupi de la comunicació entre objectes. L'objecte de control serà el que s'ocuparà de dirigir els diferents camins del cas d'ús. L' especificació de l'objecte de control s'obtindrà dels diversos camins del cas d'ús, probablement expressats en forma de diagrames d'activitat.

Les classes de control generalment no tenen atributs. Les operacions de les classes de control contenen els algorismes de l'aplicació. Entre les tasques típiques dels objectes de les classes de control hi trobem: enregistrar l'activitat del cas d'ús, cridar els controls de seguretat, etc.

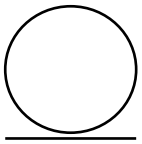
Classes de Frontera

Els objectes de les classes frontera defineixen les interfícies amb els actors: finestres, diàlegs per pantalla i menús. Generalment seran pantalles de computador. S'utilitzen també per a cridar altres subsistemes. Un cas d'ús pot tenir diversos objectes frontera, i de vegades dos o més casos d'ús poden compartir un mateix objecte frontera. N'hi ha d'haver almenys un per a cada actor.

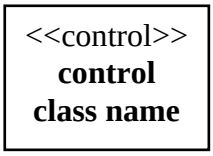
El modelat de les classes de control i de frontera s'acostuma a fer a la fase de disseny.

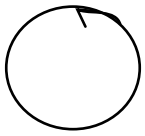
Notacions alternativa per a les classes d'anàlisi

<<entity>>
entity class name

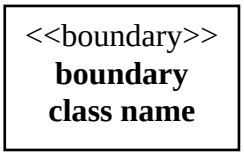


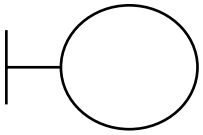
entity class name





control class name





boundary class name

Disseny estàndard de sistemes interactius

L'ampliació d'un cas d'ús fent servir classes d'anàlisi porta a una construcció estàndard dels sistemes interactius que pren una forma semblant a:

1) els actors interactuen amb els objectes frontera;
2) els objectes frontera fan peticions als objectes de control;
3) els objectes de control manipulen els objectes entitat.

A la fase de disseny, generalment, s'incorpora en forma d'operacions:

- una classe de control a una classe de frontera;
- una classe de control a una classe d'entitat;
- una classe de control a una altra classe de control;

Diagrames d'interacció.

Els **diagrames d'interacció** (de seqüència i de comunicació) són models dinàmics que descriuen com col·laboren els objectes que configuren un cas d'ús (o un escenari).

Els diagrames d'interacció modelen la interacció entre objectes, i els objectes interactuen enviant-se missatges.

Els diagrames d'interacció es dibuixen a les fases d'anàlisi i sobretot a la de disseny, amb el propòsit d'ajudar a decidir les operacions de cada classe.

Les responsabilitats d'un objecte en quan al seu comportament poden ser de dos tipus:

- Fer:
  - sobre ell mateix: fer un càlcul;
  - iniciar una acció sobre altres objectes;
  - controlar i coordinar activitats entre objectes.
- Conèixer:
  - dades privades encapsulades;
  - dades d'objectes relacionats;
  - el que pot derivar o calcular.

Els mètodes d'una classe s'implementen per tal que els objectes de la classe puguin complir les seves responsabilitats. Les responsabilitats s'assignen als objectes durant la fase de disseny.

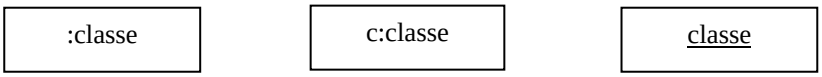
Els diagrames d'interacció ajuden a determinar els mètodes que caldrà implementar a les diferents classes per a que els seus objectes siguin capaços de complir amb les seves responsabilitats.

Recordeu que:

- un objecte es representa amb un rectangle;
- el nom de la classe de l'objecte va precedit de (:);
- es pot donar explícitament o no el nom de l'objecte;
- el nom de l'objecte:classe es subratlla.



tot i que moltes vegades, per abús de llenguatge, només es posa:



Per a invocar un missatge a un diagrama d'interacció s'acostuma a fer servir la notació:

returnValue:=operName(param1:paramType1, ..., paramN:paramTypeN): returnType

La presència d'un missatge a un diagrama d'interacció implica l'existència d'una operació, amb la mateixa signatura que el missatge, a la classe del objecte receptor del missatge.

El conjunt de tots els missatges a objectes d'una classe pel conjunt de tots els diagrames d'interacció de tots els casos d'ús d'una aplicació determina els mètodes que ha de tenir aquesta classe.

La presència d'un missatge a un diagrama d'interacció implica l'existència d'una associació en el diagrama de classes entre les classes dels objectes que interactuen per mitjà del missatge.

Els diagrames d'interacció es fan servir per a:

- Validar la lògica dels casos d'ús;
- Guiar el procés d'assignar funcionalitats a les classes de disseny a partir de les operacions que surten en els diagrames;
- Mostrar les classes de disseny que tindran operacions complexes. Per tant, probablement per aquestes operacions caldrà construir diagrames d'estat.

Els diagrames de seqüència i els diagrames de comunicació són equivalents: proporcionen vistes alternatives de la mateixa informació.

Cal tenir clar que si en lloc de modelar un flux de missatges el que es vol és modelar un flux de dades o de processos, el que cal utilitzar és un diagrama d'activitats.

Recomanacions generals sobre els diagrames d'interacció:

- crear diagrames especialment per a les interaccions més importants;
- donar nom als objectes quan es referencien en els missatges;
- donar nom als objectes quan en el diagrama existeixen diversos objectes del mateix tipus;
- posar els paràmetres només quan no són prou clars;
- és millor posar el nom dels paràmetres en comptes del tipus;
- no modelar el valor de retorn quan és obvi el valor que retorna un missatge;
- modelar un valor de retorn d'un missatge si posteriorment ha de ser utilitzat;
- modelar els valors de retorn d'un missatge com a part de la invocació d'un missatge;
- a l'enviar un missatge a un objecte o una classe, cal assegurar-se que a la classe existeix la corresponent operació, que ha de ser de tipus static si és operació de classe.
- Els diagrames d'iteració no són adients per a fer el modelat detallat d'operacions. Aquest modelat es fa millor fent servir pseudocodi (o diagrames d'estat).

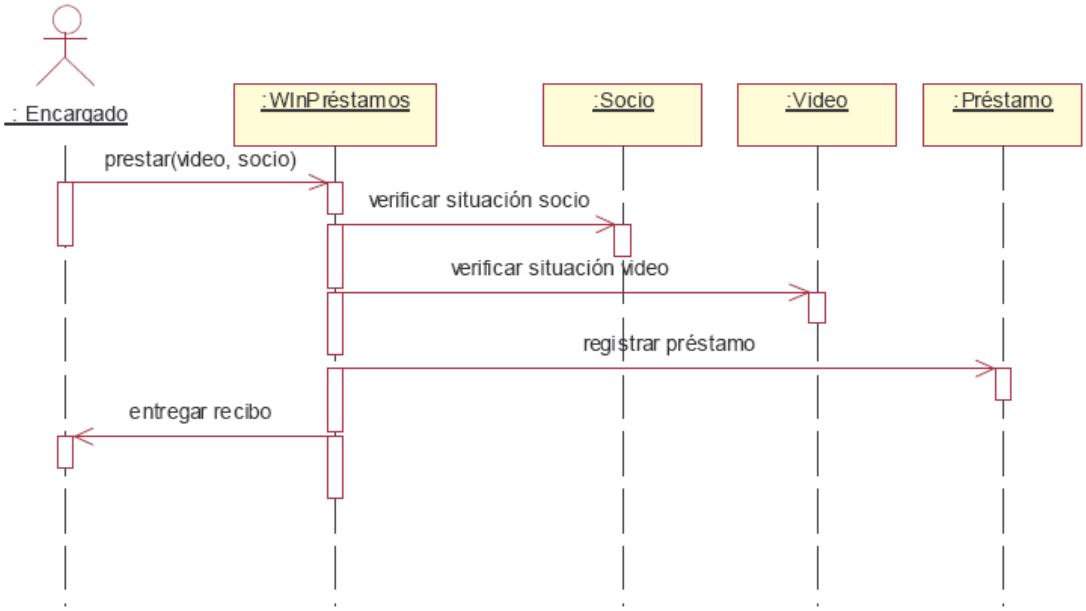
Diagrames de seqüència

Un **diagrama de seqüència** és un diagrama d'interacció que detalla com s'executen les operacions en funció del temps: quins missatges són enviats, per quin objecte, a quin objecte i quan.

Els diagrames de seqüència són molt útils per a modelar escenaris.

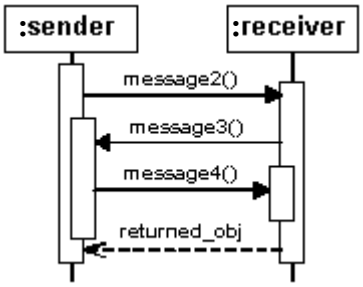
Els objectes involucrats en el diagrama es llisten d'esquerra a dreta d'acord amb el moment que intervenen en la seqüència de missatges.

A continuació donem l'exemple d'un diagrama de seqüència, a molt alt nivell, per al cas d'ús fer un préstec d'un vídeo a un soci d'una videoteca:

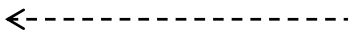


Cada línia vertical puntejada és una *línia de vida* que representa el temps durant el qual un objecte existeix. Cada fletxa representa una crida a un missatge. Una fletxa va des de l'emissari fins a la part superior de la *barra d'activació* del missatge situada a la línia de vida del receptor. La barra d'activació representa el temps d'execució del missatge.

A continuació donem un altre exemple genèric, per a mostrar com es fan servir les barres d'activació:

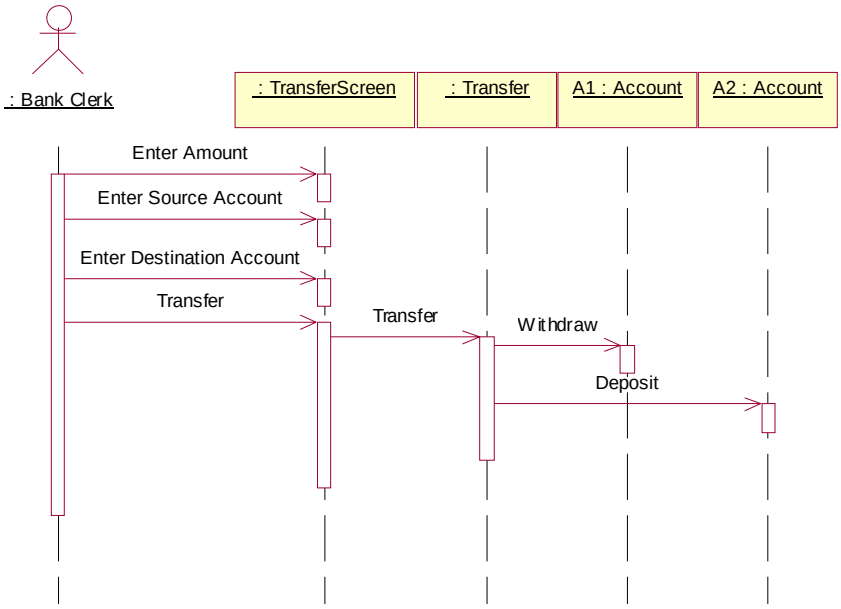


Els valors de reton s'indiquen (opcionalment) fent servir una fletxa puntejada:



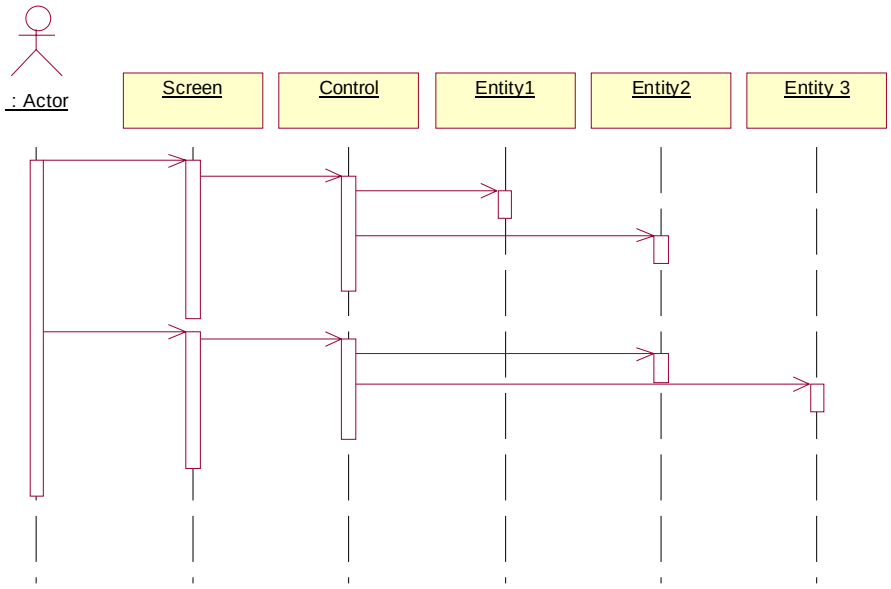
A l'exemple anterior es pot veure que per a indicar que una nova operació d'un objecte ha estat invocada com a part de l'execució d'una operació prèvia, les barres d'activació es nien.

A continuació donem l'exemple d'un diagrama de seqüència que correspon al cas d'ús transferir diners d'un compte a un altre que hem vist anteriorment:



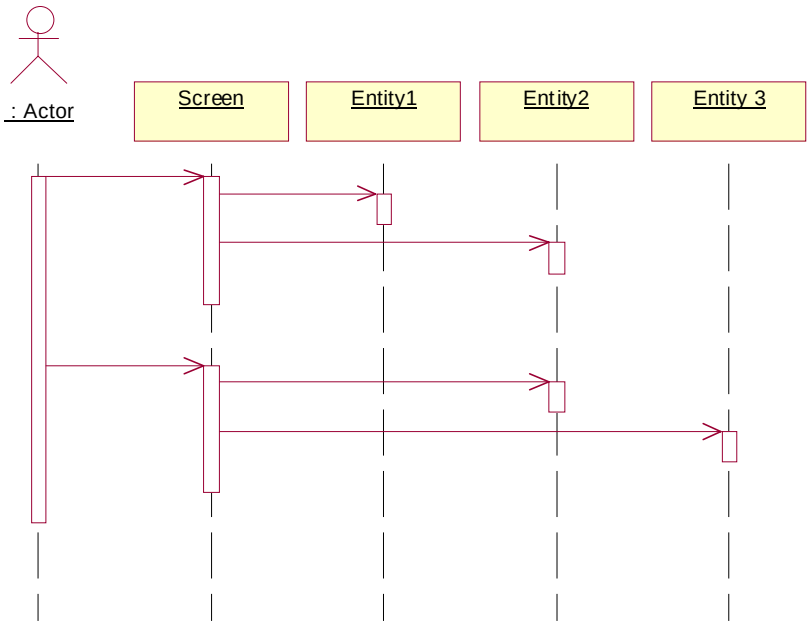
Disseny estàndard de diagrames de seqüència per a sistemes interactius

L'ampliació d'un cas d'ús fent servir classes d'anàlisi porta a un disseny estàndard dels sistemes interactius que pren una forma semblant a:



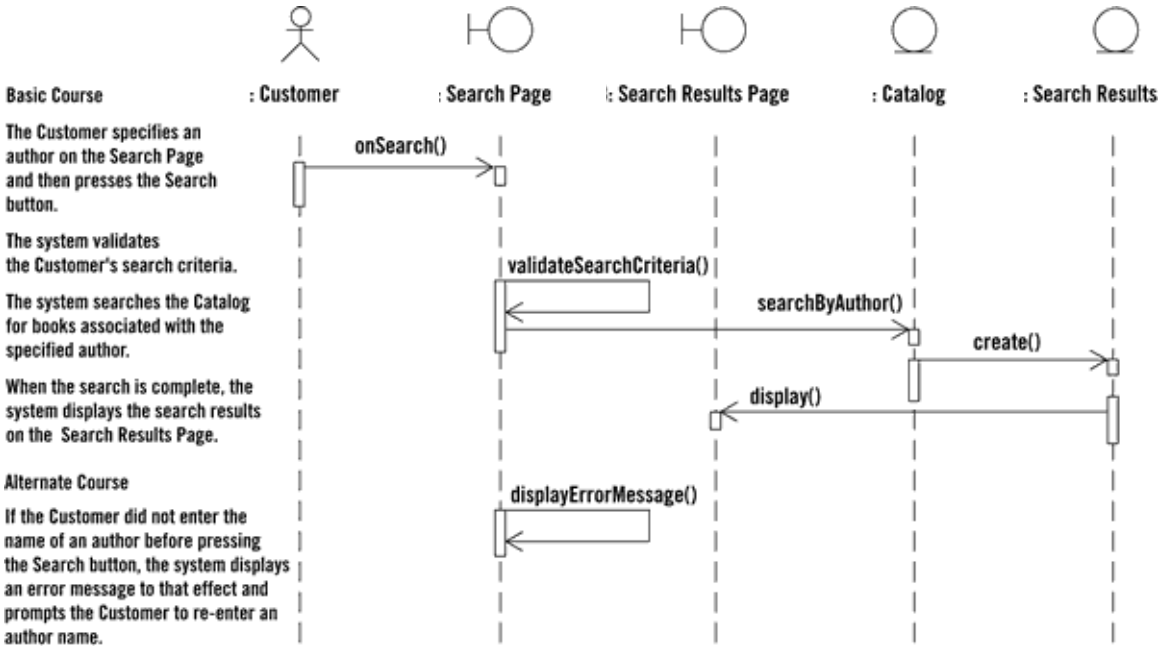
L'objecte de la classe de control se n'encarrega de gestionar el cas d'ús que s'està modelant. S'ocupa de la lògica (coordinació i control de l'activitat) del cas d'ús. Generalment delega la realització del treball a altres objectes.

A la fase inicial de l'anàlisi, per tal de simplificar els diagrames, de vegades s'inclouen els objectes de control dins dels objectes de frontera i a les hores obtenim un diagrama de seqüència simplificat de la forma:



Ara els actor interactuen amb la pantalla i la pantalla manipula els objectes entitat.

A continuació donem altres exemples de diagrames de seqüència:



Requadres de diagrames en els diagrames de seqüència UML

Per a suportar diferents tipus de constructors, el UML fa servir **requadres**. Els requadres són regions o fragments de diagrames. Els requadres tenen un operador o etiqueta i una clàusula condicional.

A continuació un exemple d'un requadre:

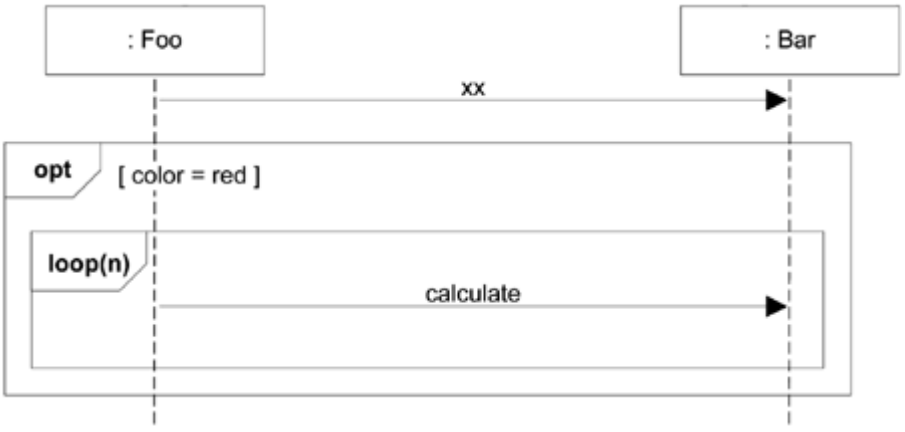


Opcionalment, tot diagrama de seqüència pot ser bordejat amb un requadre amb l'etiqueta **sd** (**s**eqüence **d**iagram), seguit del nom donat al diagrama.

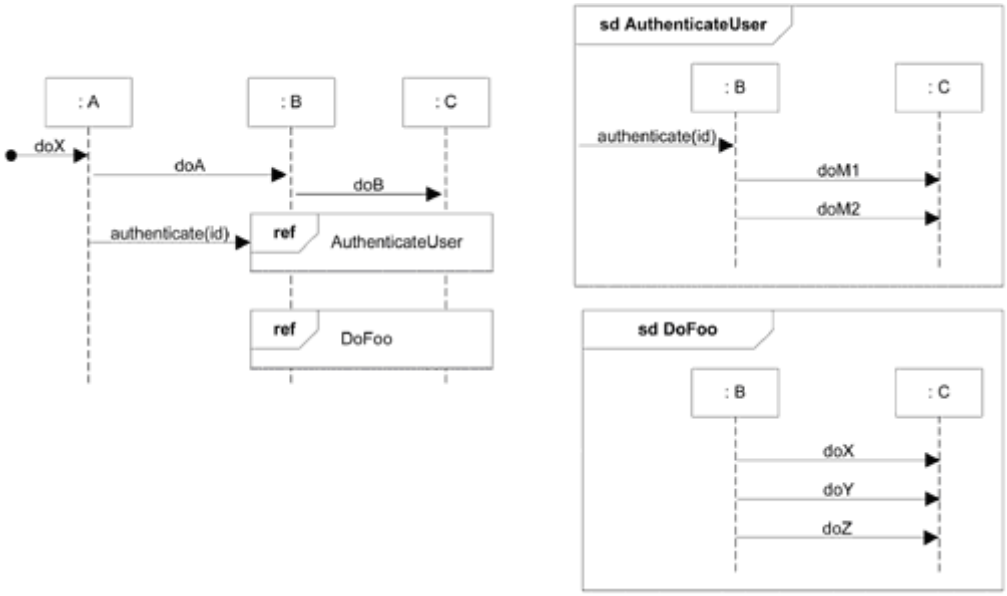
La següent taula resumeix alguns dels operador de requadre més comuns.

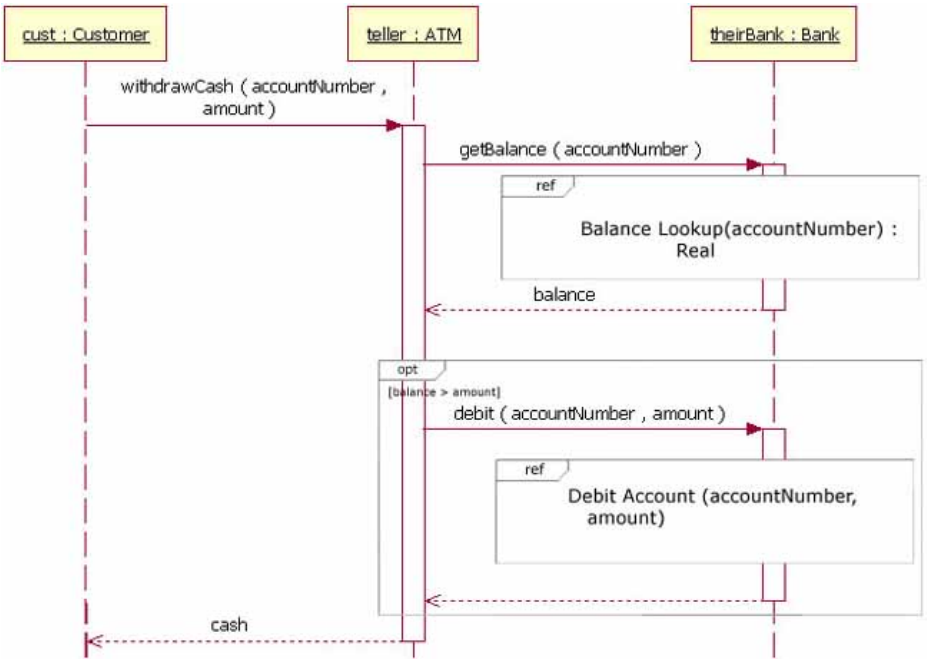
Operador	**Significat**
alt	El fragment alternatiu per a una exclusió mútua determinada per la condició lògica expressada en el condicional.
break	Es molt semblant al fragment opcional, excepte que si compleix amb la condició de break executa les instruccions del requadre de break i acaba, es a dir, la resta dels missatges posteriors al break no són executats.
loop	El fragment recursa mentre la condició establerta sigui verdadera. Es pot escriure *loop(n)* per a indicar que el fragment es repeteix *n* vegades.
opt	El fragment opcional que s'executa si la condició es verdadera.
par	Els fragments s'executen en paral·lel.
ref	Permet referenciar altres diagrames de seqüència per tal de reutilitzar diagrames existents o simplificar diagrames complexos. El diagrama al que fa referència ha de posar-se en un requadre etiquetat amb **sd** i la signatura de la operació de crida o el nom del diagrama.

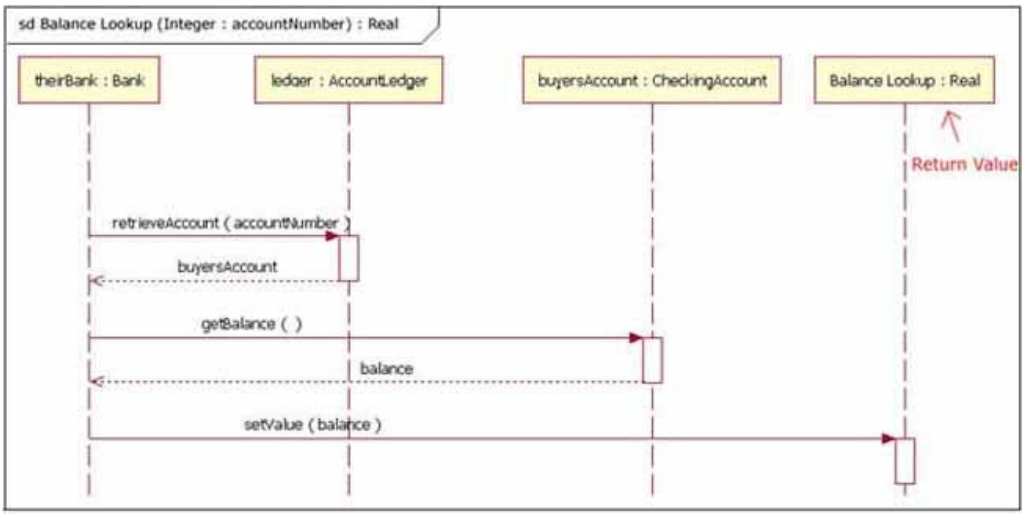
Els fragments poden estar aniats uns en altres com es pot apreciar al diagrama següent:



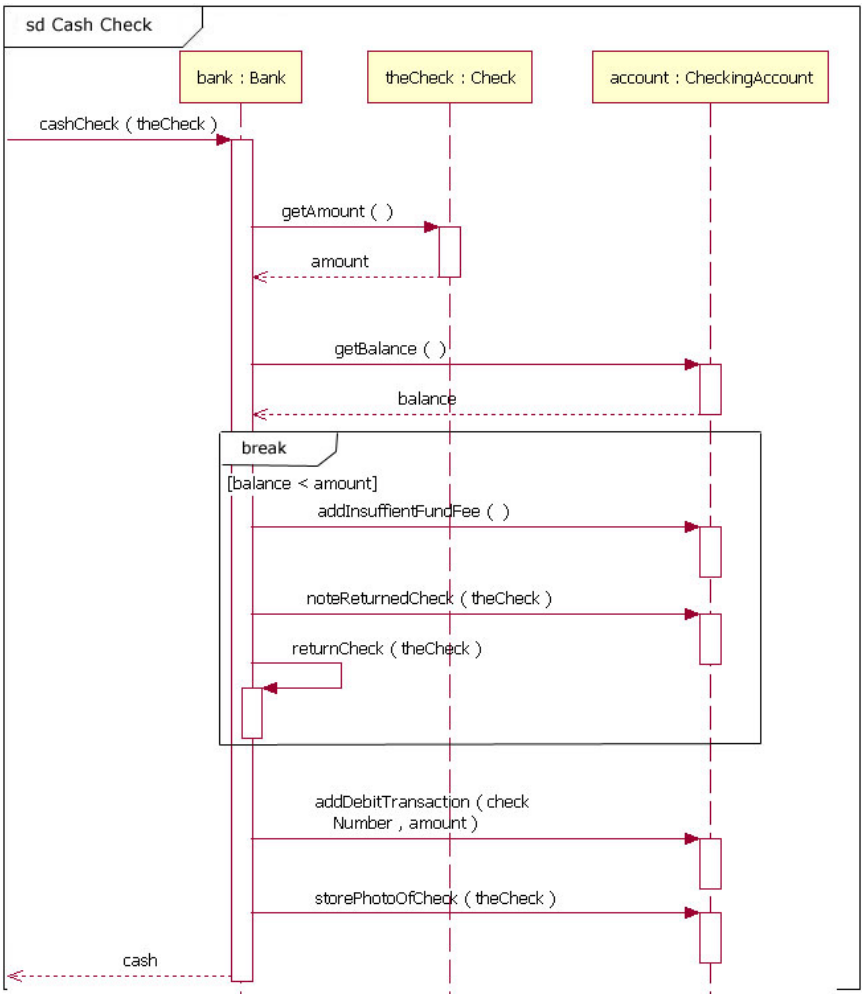
A continuació alguns exemples de com relacionar diagrames de seqüència.

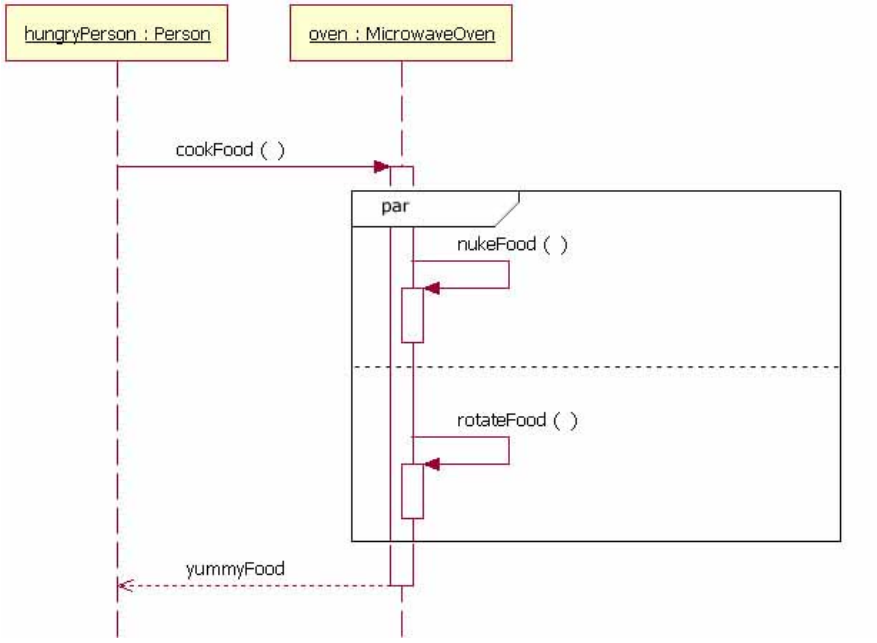






Exemples de fragments **break** i **paral·lel**:




hungryPerson : Person
oven : MicrowaveOven
cookFood ( )
par
nukeFood ( )
rotateFood ( )
yummyFood

Diagrames de comunicació

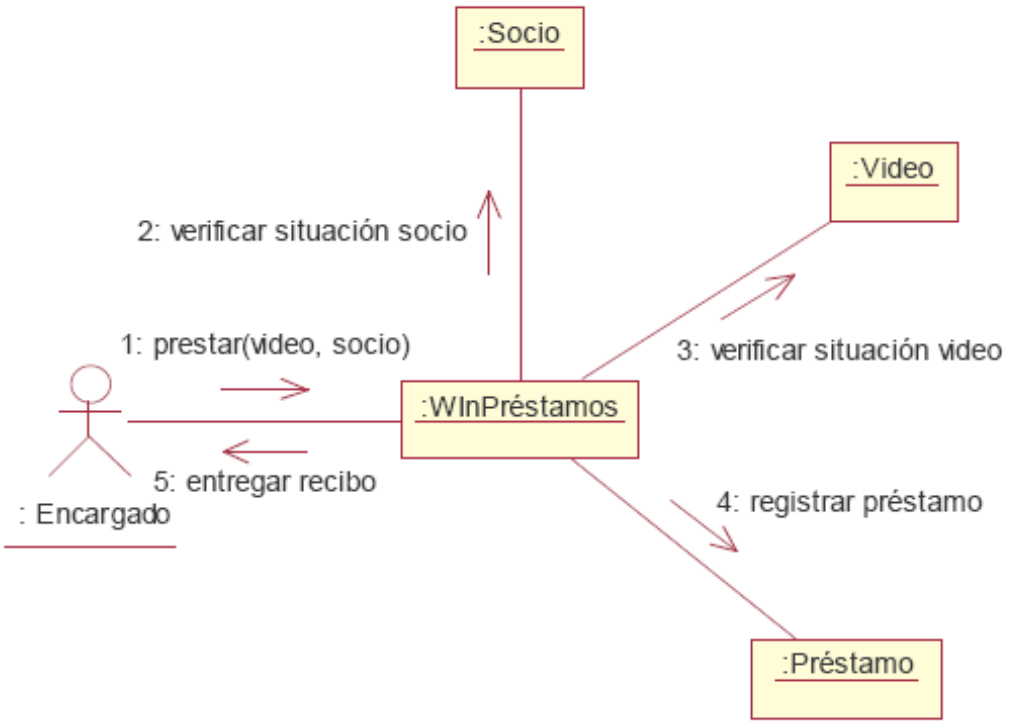
Els diagrames de comunicació proporcionen la mateixa informació que els diagrames de seqüència, però centrant-se en els rols del objectes en comptes de fer-ho en el temps en el qual s'envien els missatges.

Un lligam (link) és una connexió entre dos objectes que instancia una associació entre les classes corresponents. Es representa per mitjà d'un segment dibuixat entre els objectes. A un mateix lligam li poden correspondre múltiples missatges i missatges en les dues direccions. La direcció de cada missatge s'indica amb una petita fletxa situada al costat del missatge.

Cada missatge d'un diagrama de comunicació té un nombre de seqüència. El missatge inicial és el nombre 1. Els missatges d'igual nivell (enviats a la mateixa crida) tenen com a prefix el mateix nombre, seguit d'un sufix 1, 2, etc. D'acord amb el moment que passa.

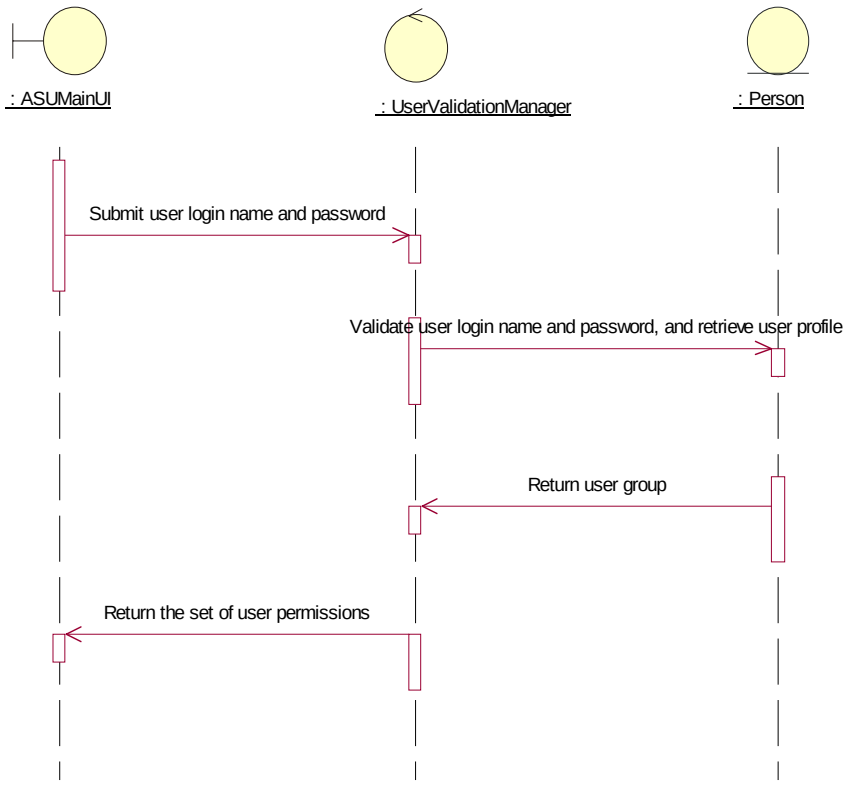
Exemples:

A continuació donem l'exemple d'un diagrama de comunicació per al cas d'ús fer un préstec d'un vídeo a u soci:

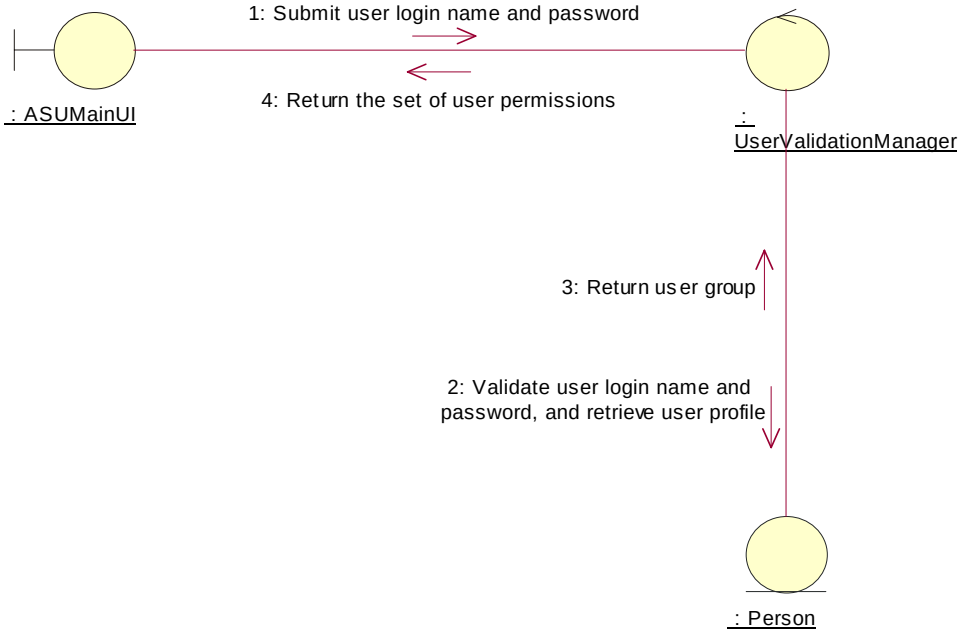


A continuació donem un parell d'exemples de diagrames de seqüència i comunicació equivalents:

D. S.



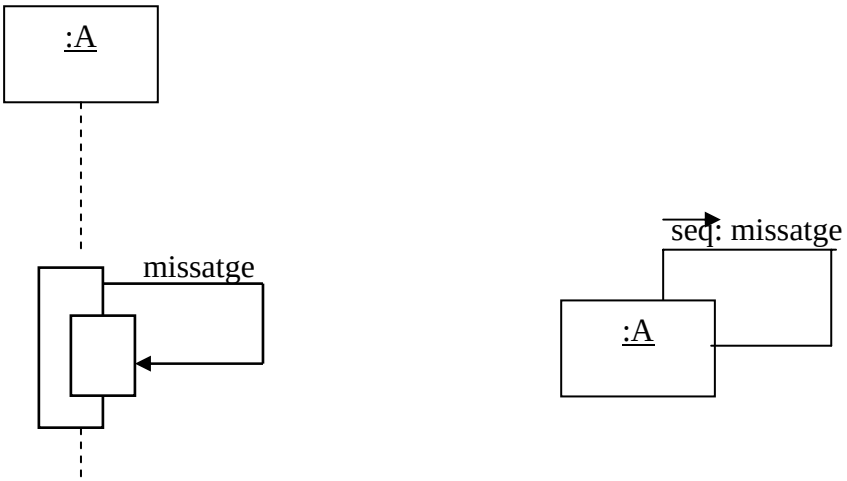
D. C.



Missatges i diagrames d'interacció

Missatge d'autocrida

La representació d'una autocrida (missatge que un objecte s'envia a ell mateix) a un diagrama de seqüència i a un diagrama de comunicació es representa per:



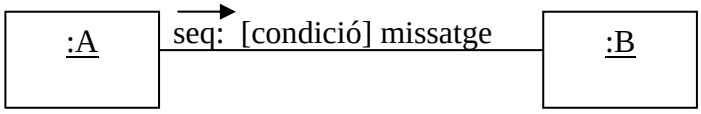
Missatge Condicional

La sintaxi d'un missatge condicional és:

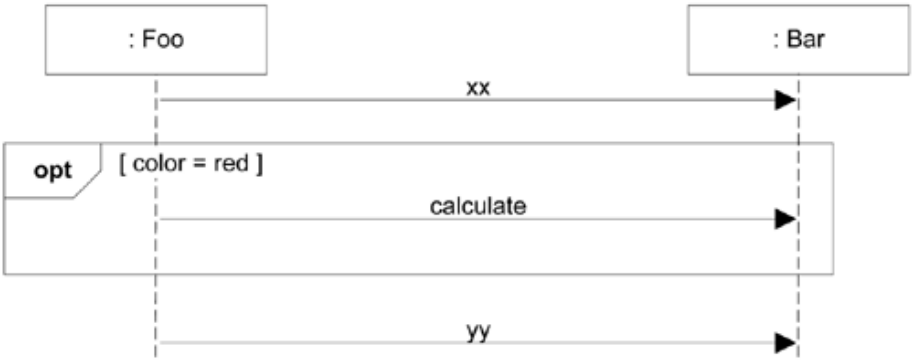
[condició] missatge

El missatge s'envia només si la condició és certa.

Exemple de missatge condicional en un diagrama de comunicació:

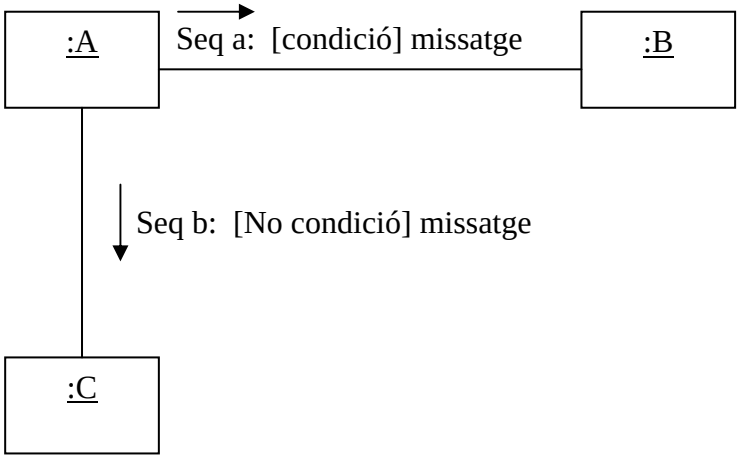


En el cas d'un diagrama de seqüència, el condicional s'expressa fent servir un fragment opcional etiquetat amb opt i seguit de la condició:

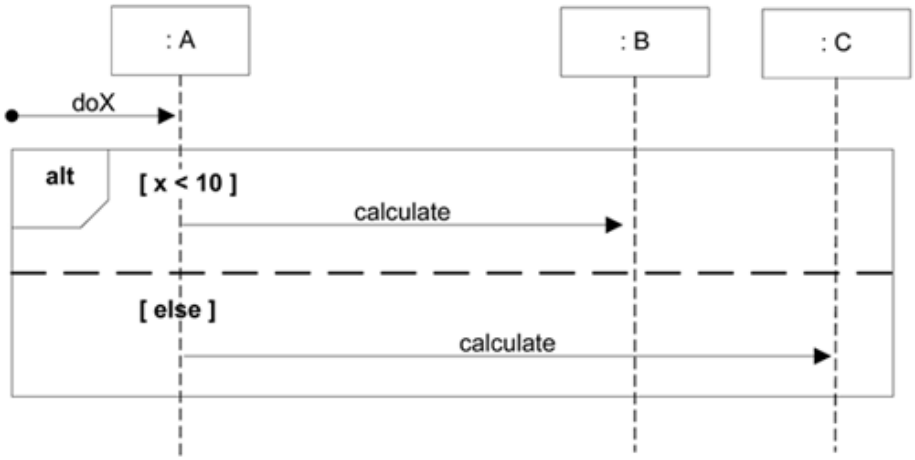


Missatges Excloents

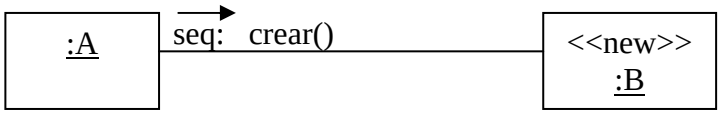
Exemple de missatges excloents a un diagrama de comunicació:



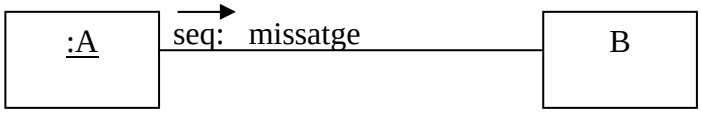
En un diagrama de seqüència els missatges excloents apareixen dins d'un requadre **alt**:



Missatge de creació d'instància

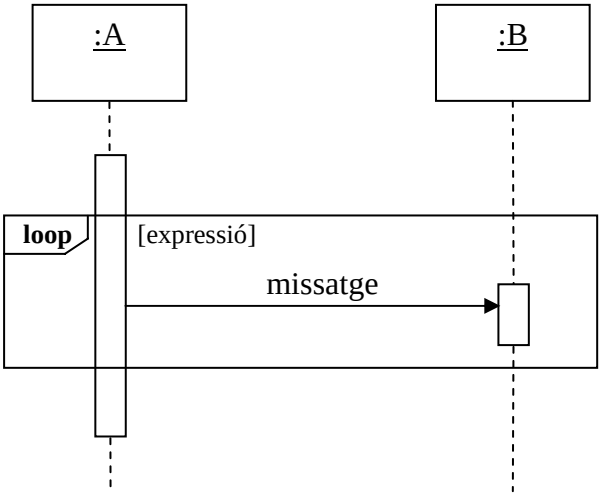


Missatge a una classe



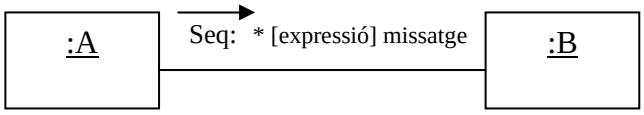
Iteració de missatges

La representació d'una iteració a un diagrama de seqüència es fa de la forma següent:



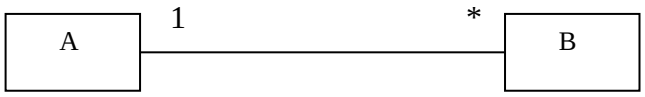
La sintaxi per a indicar que un missatge s'envia repetidament a un mateix objecte a un diagrama de comunicació és:

* [expressió] missatge



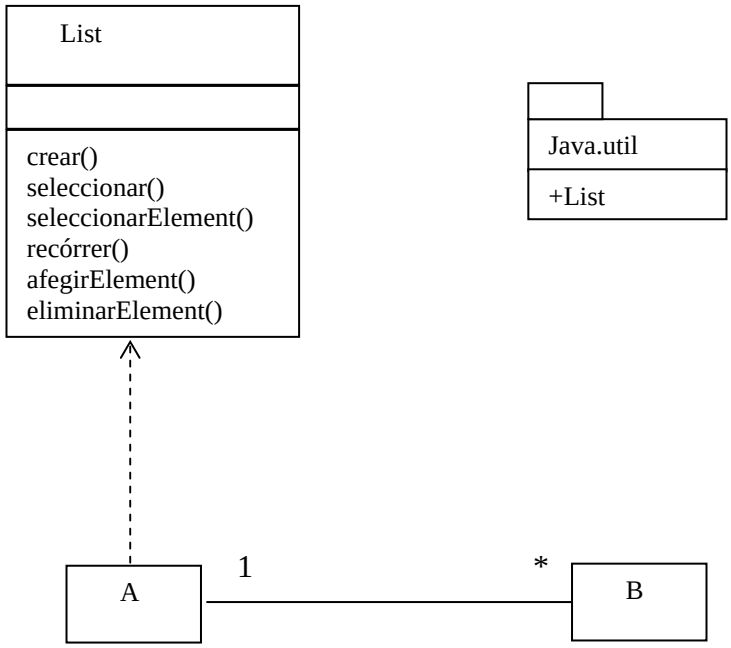
Col.leccions

Quan tenim una associació entre dues classe A i B de la forma:

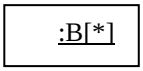


significa que tot objecte de la classe A té una referència a una *col·lecció* d'objectes de la classe B.

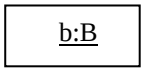
Quan es treballa amb Java, la col·lecció és una classe del paquet *java.util*, per exemple: *List*, *Map*, ..., que acostuma a tenir mètodes del tipus: *crear()*, *buscar()*, *selecionar()*, *seleccionarElement()*, *recórrer()*, *afegirElement()*, *eliminarElement()*, … . El següent diagrama de classes ens mostra amb detall aquesta situació:



Una col·lecció d'objectes de la classe B, considerada com una unitat, es simbolitza per :



i un objecte específic b de la col·lecció B[*], per:



Quan un missatge s'envia a tota una col·lecció d'objectes, el missatge és part de la interfície de la col·lecció, i no pas de la interfície dels seus membres. Exemples:

Diagrama de comunicació:

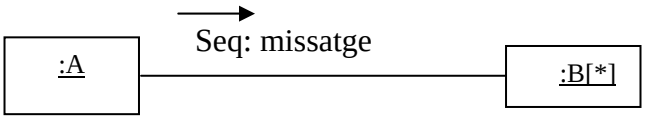
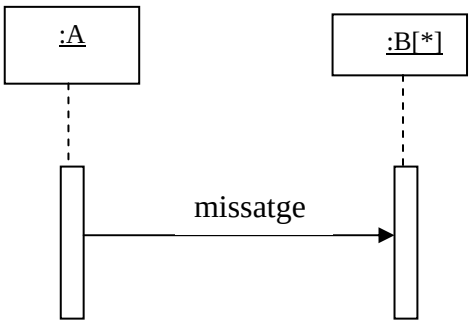


Diagrama de seqüència:



Quan es crea un objecte de la classe A, aquest s'ha d'ocupar de crear una col·lecció (buida) d'objectes de la classe B, que queda referenciada per l'objecte de la classe A:

Diagrama de comunicació:

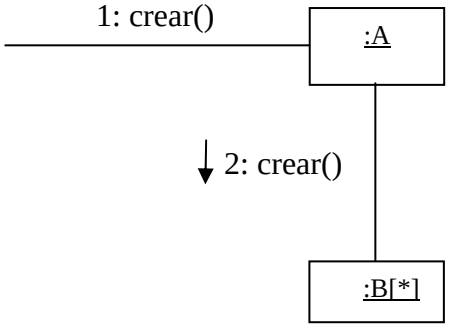
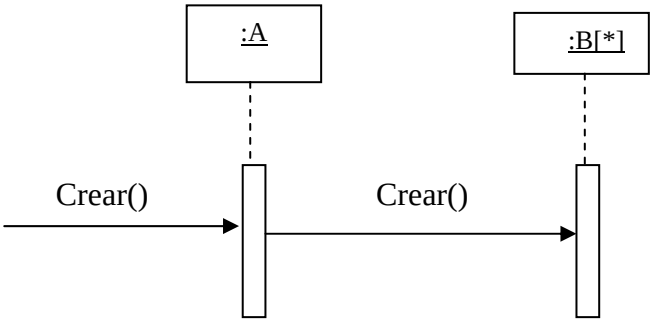
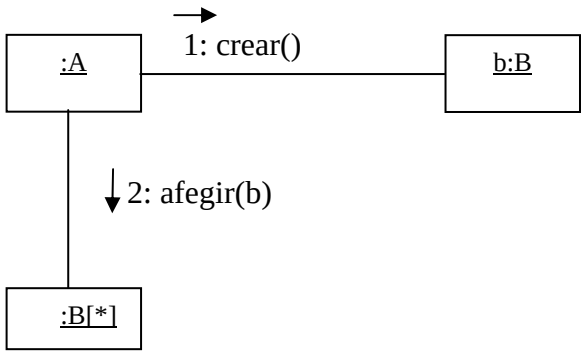


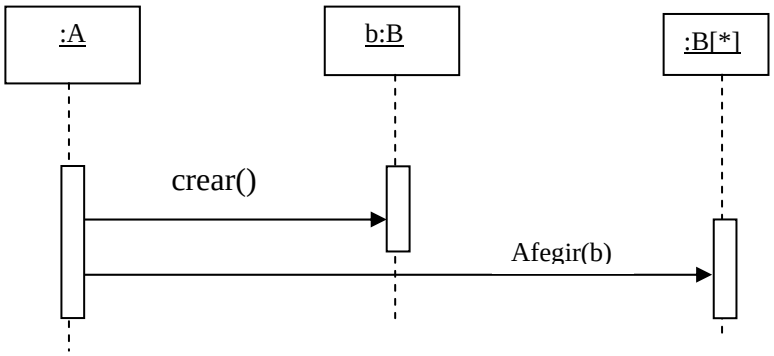
Diagrama de seqüència:



El diagrama de comunicació que segueix a continuació, mostra la creació d'un objecte de la classe B, que posteriorment és afegit a la col·lecció d'objectes de la classe B:



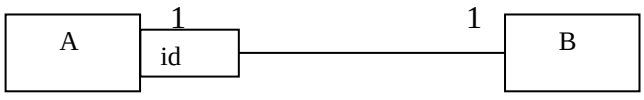
El diagrama de seqüència del mateis exemple seria:



Per a realitzar una operació sobre determinats objectes d'una col·lecció calen dos missatges:

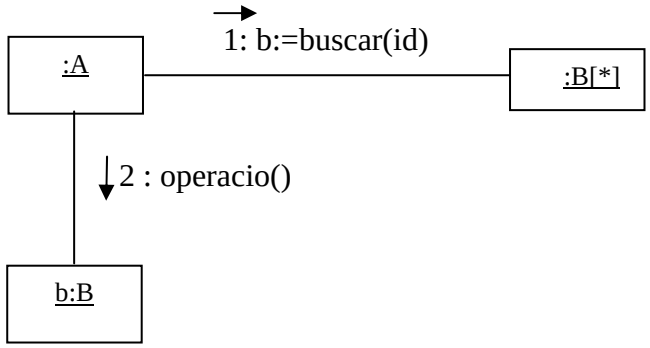
- un primer missatge sobre la col·lecció per a extreure'n els objectes individuals desitjats;
- un segon missatge enviat a cada objecte individual, fent servir el lligam temporal creat pel primer missatge.

A continuació donem un exemple d'un diagrama de comunicació, derivat de l'associació qualificada:

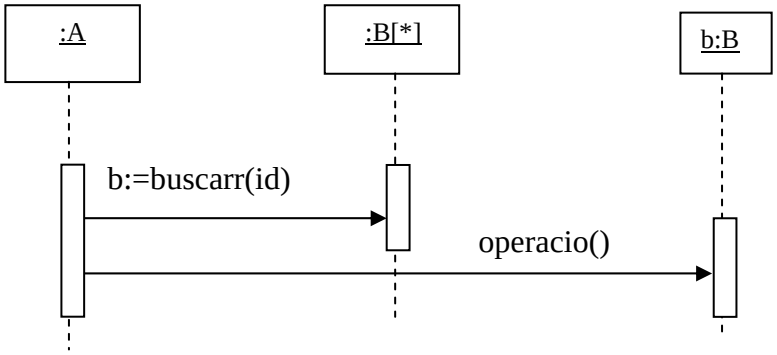


en el qual es mostra com:

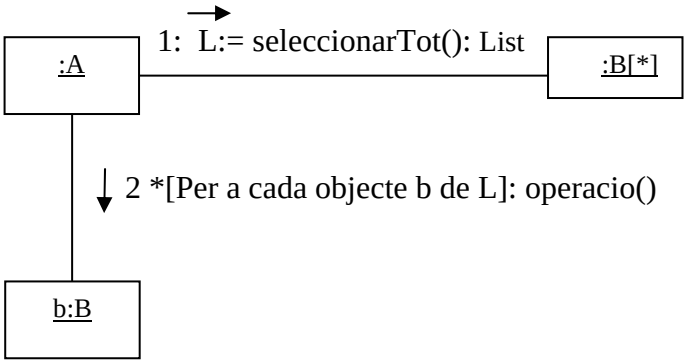
- primer es busca un objecte b:B de la col·lecció d'objectes de la classe B per mitjà de l'identificador *id*;
- després se efectua una operació sobre l'objecte b:B.



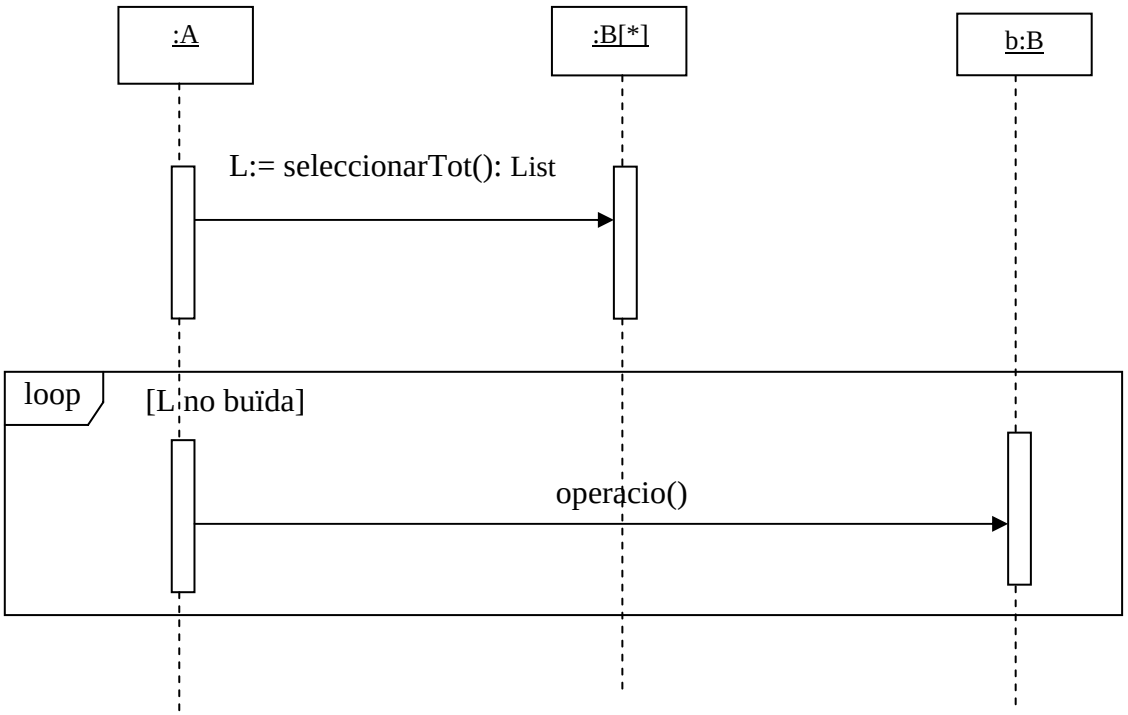
El corresponent diagrama de seqüència seria:



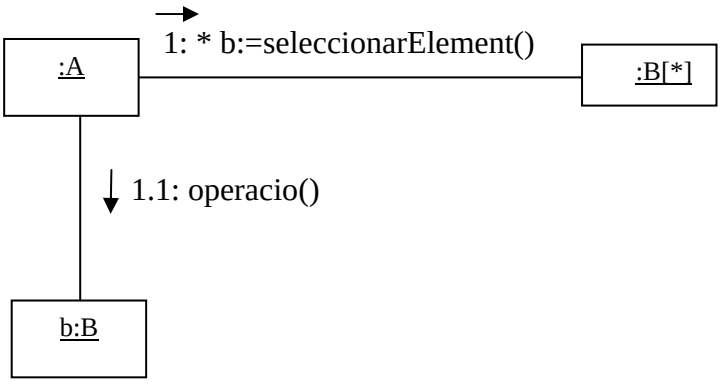
El resultat d'un missatge pot ser una col·lecció. Un exemple típic d'aquesta situació és l'operació *seleccionarTot*. La selecció de tots els elements d'una col·lecció, seguit del missatge *operacio()* sobre cadascun dels objectes del conjunt seleccionat, es representa en un diagrama de comunicació de la forma:



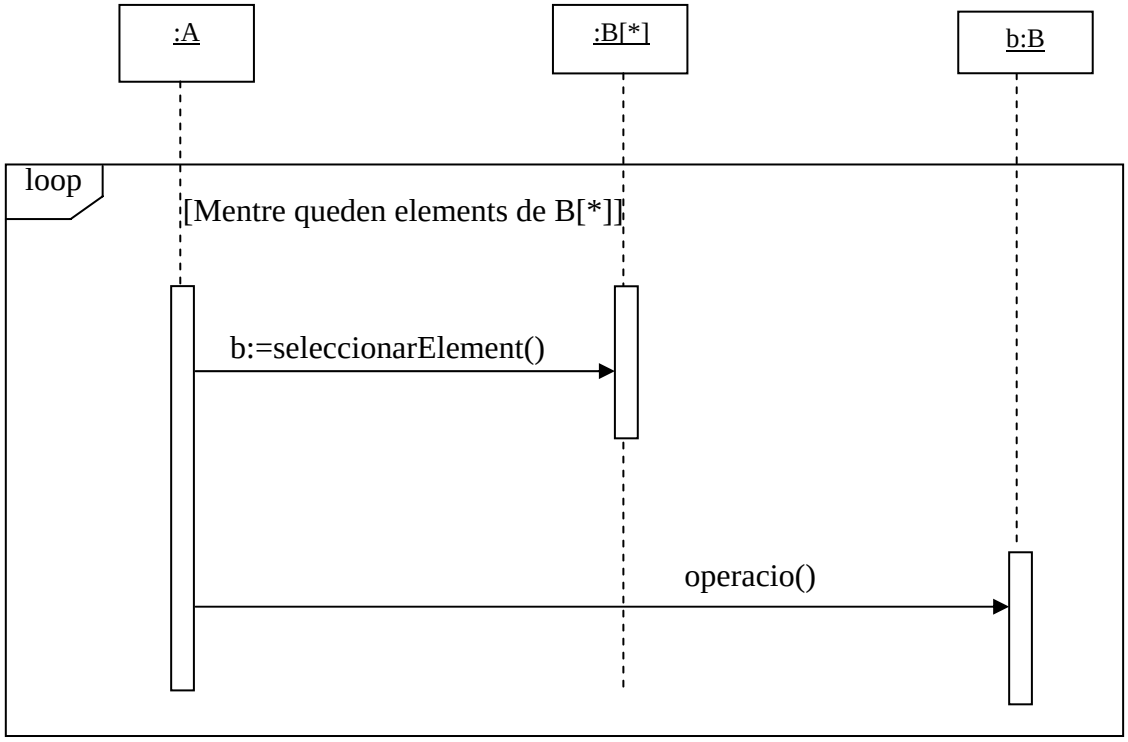
I en un diagrna de seqüència:



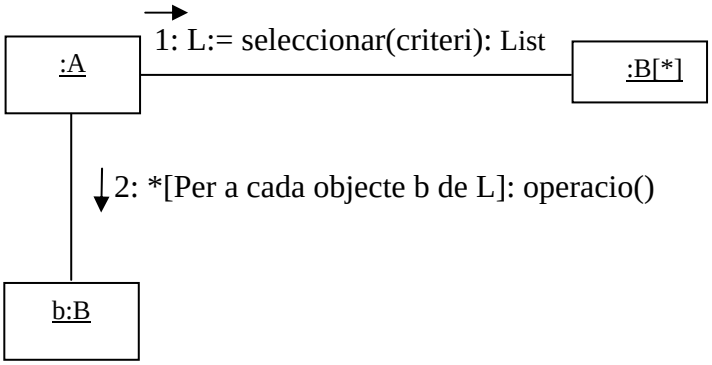
Quan s'està fent un recorregut seqüencial d'una col·lecció, el missatge *seleccionarElement()* adreçat a la col·lecció retorna el primer objecte no trobat. Per a indicar que hi ha un recorregut sobre tots els objectes de la col·lecció, de manera que a mesura que es van trobant s'envia el missatge *operacio()* a cadascun dels objectes seleccionats, podem fer servir un diagrama de comunicació:



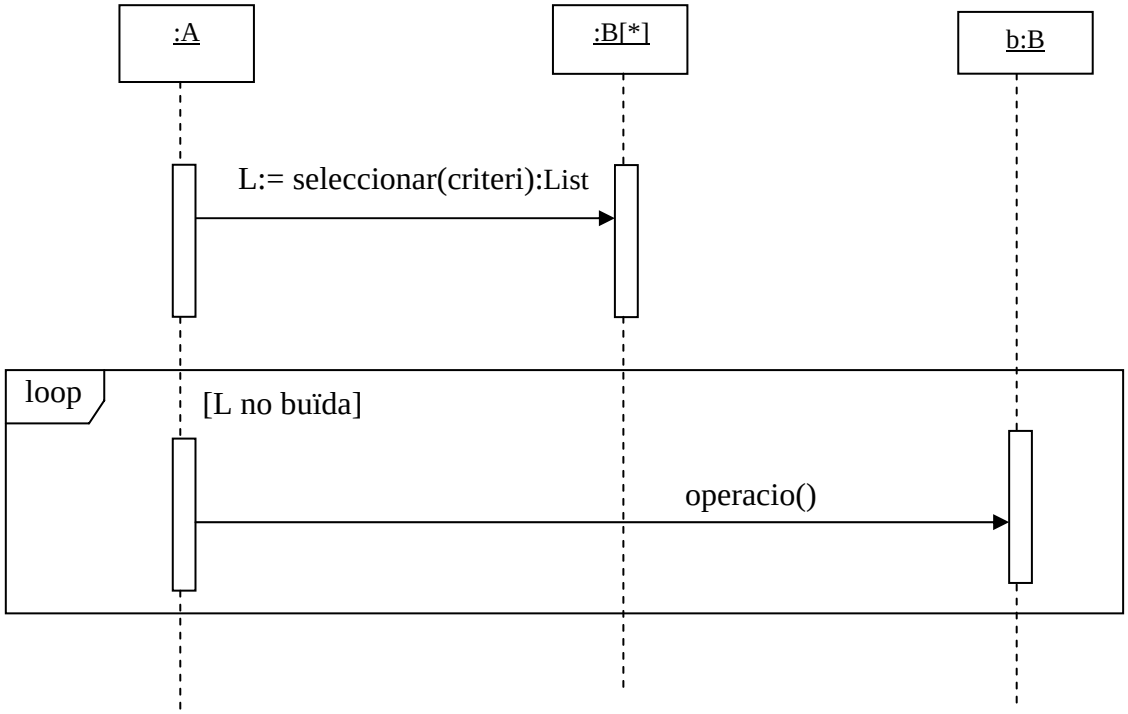
o be un diagrama de seqüència:



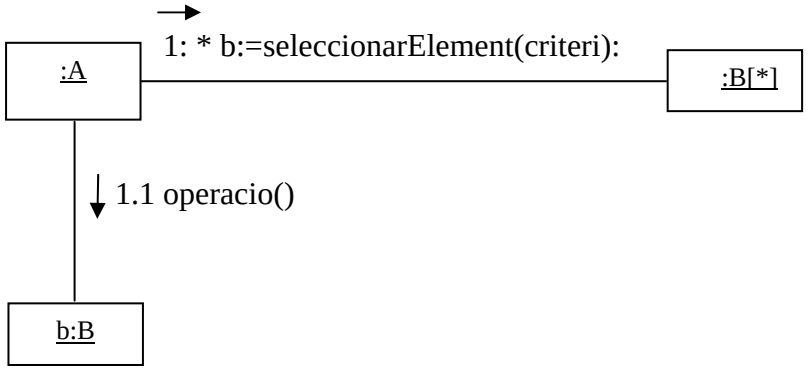
A continuació donem un exemple de diagrama de comunicació, amb un missatge seleccionar segons criteri sobre una col·lecció, seguit del missatge *operacio()* enviat a cadascun dels objectes de la llista seleccionada:



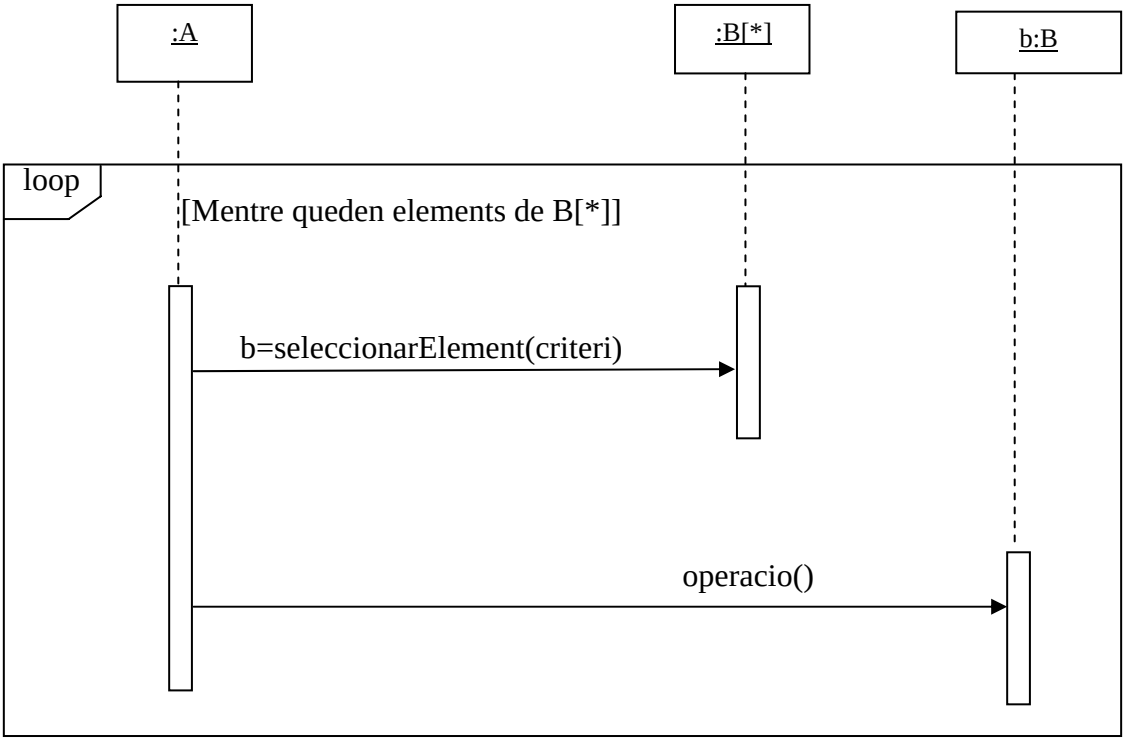
El corresponent diagrama de seqüència seria:



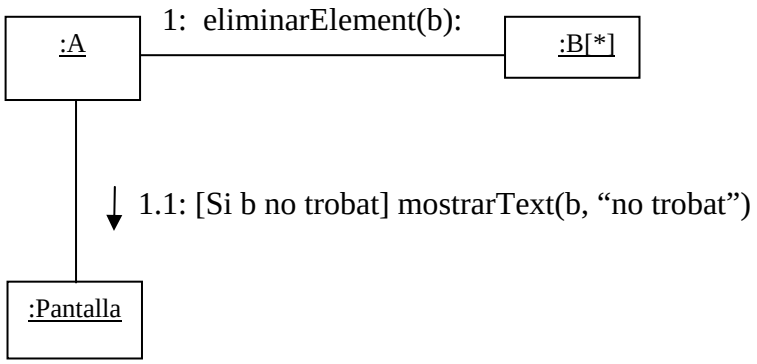
Quan s'està fent un recorregut seqüencial d'una col·lecció, el missatge *seleccionarElement(criteri)* adreçat a la col·lecció retorna el primer objecte no trobat que compleix *criteri.* El següent diagrama de comunicació ens indica que hi ha un recorregut sobre els objectes d'una Col·lecció, de manera que s'envia el missatge *operacio()* a cadascun dels objectes b:B de la Col·lecció que compleixen *criteri*, a mesura que es van trobant:



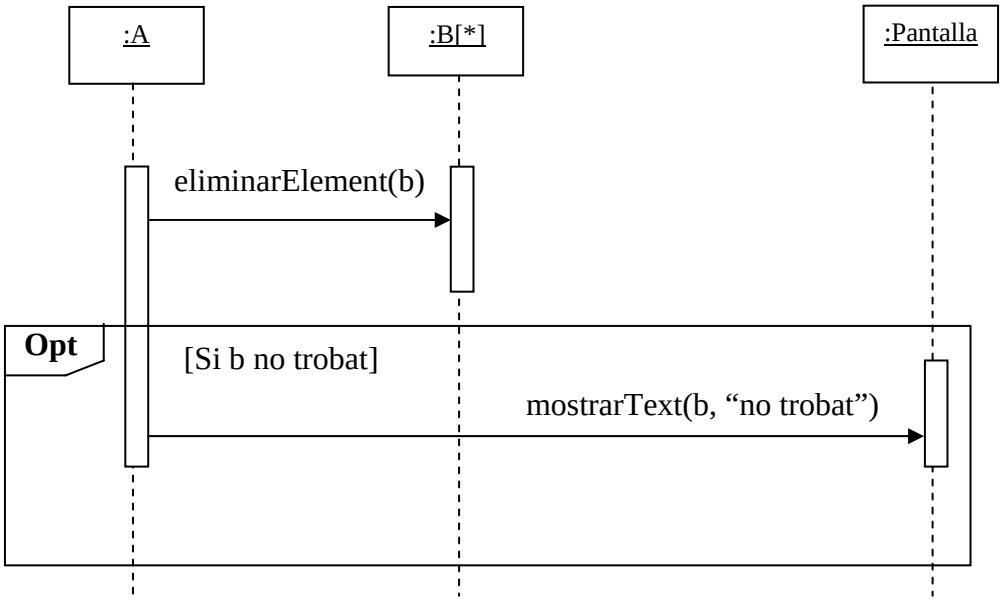
I el diagrama de seqüència de la mateixa situació seria:



Una col·lució pot enviar un missatge a un objecte. El diagrama de comunicació de l'exemple següent mostra com, al no poder-se executar una ordre d'eliminar un objecte d'una col·lecció per no existir, la col·lecció envia un missatge a la pantalla per a indicar que l'objecte no s'ha trobat:



El corresponent diagrama de seqüència seria:



APÈNDIX.

DELS DIAGARAMES DE CLASSES I DE COMUNICACIÓ AL CODI JAVA

La Figura 1 mostra el diagrama de classes d'una aplicació en la qual hi inetrvenen les classes: Company, Store, Order i Delivery.

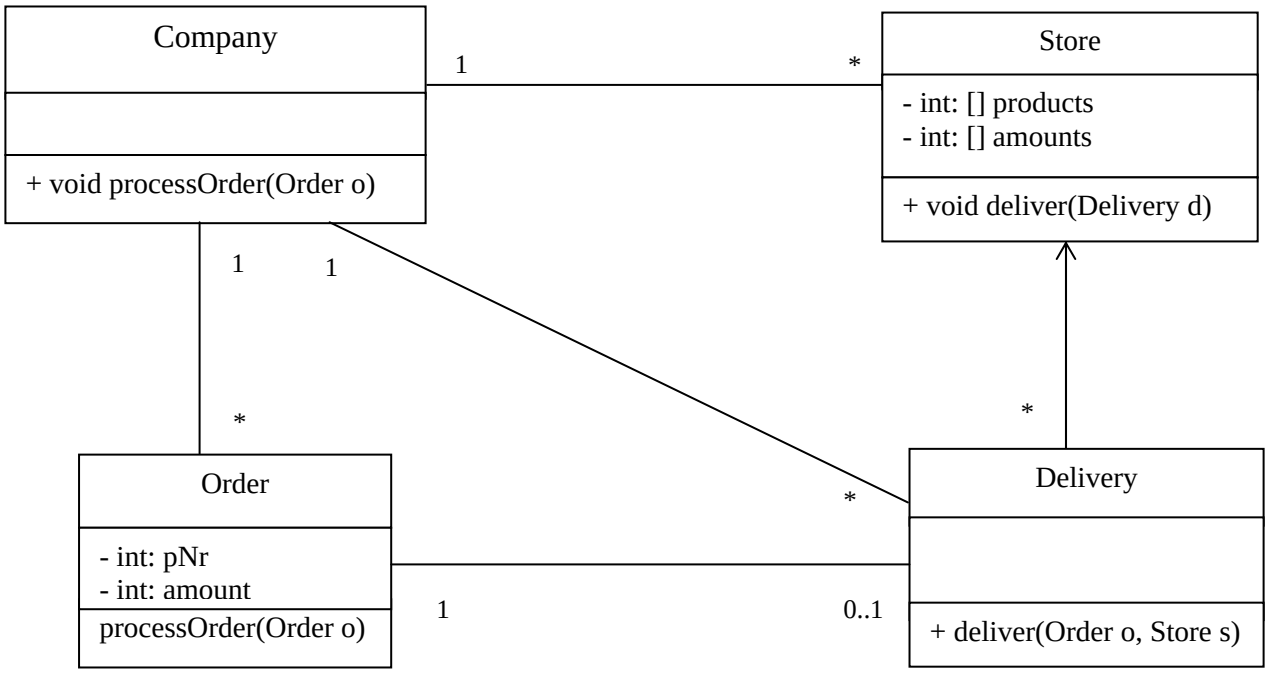


Figura 1

La Figura 2 mostra un diagrama de comunicació de l'operació processOrder: l'objecte Company obté el nombre de producte *pNr* i la quantitat encarregada *amount* del producte fent servir les corresponents funcions d'accés de la classe Order. Després comprova entre tots els objectes Store relacionats, quin pot suministrar la quantita encarregada del producte. Es crea un objecte Delivery i l'objecte Store seleccionat s'ocupa d'enviar-la. Finalment, l'objecte Delivery s'afegeix a la col·lecció de Deliberys de Company.

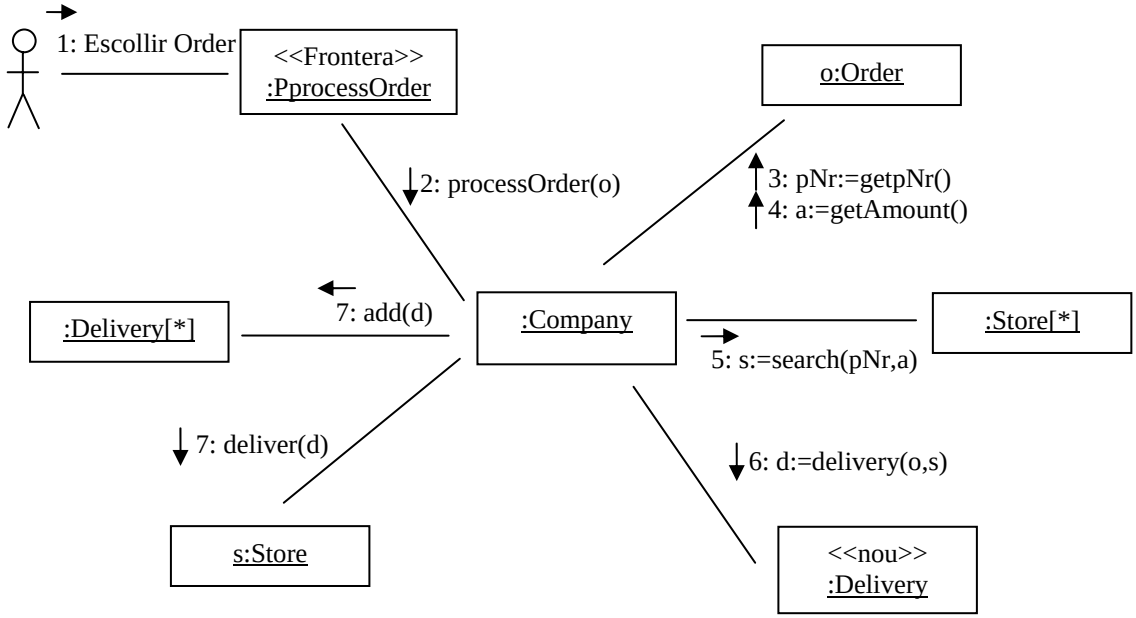


Figura 2

A continuació mostrem el codi Java que correspon al diagrama de comunicació de la Figura 2.

```
public class Company {
        ...
        public void processOrder (Order o) {
           Delivery d;
           Store s;
           int pNr;
           int a;
           pNr = o.getpNr();
           a = o.getAmount();
           s = search_stores(pNr, a);
           d = new Delivery(o,s);
           s.deliver(d);
           add_deliveries(d);
        }
        ...
}
```