

# LOG210 SÉANCE #10

## ANALYSE ET CONCEPTION DE LOGICIELS



1. Affinement du MDD ← 25.42m
2. Diagramme d'interaction
3. Diagramme d'état
4. Diagramme d'activité

# AFFINEMENT DU MODÈLE DU DOMAINE

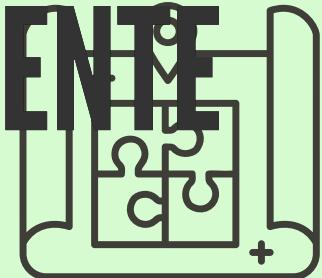


Catégorie	Exemples
Objets physiques ou tangibles	CarteDeCrédit, Chèque
Transactions	PaiementEnEspèces, PaiementACrédit, PaiementParChèque
Systèmes externes	ServiceDAutorisationDesCrédits, ServiceDAutorisationDesChèques
Organisations	ServiceDAutorisationDesCrédit, ServiceDAutorisationDesChèques, Documents de travail comptables, contractuels, juridiques

Tableau F26.1/A32.1

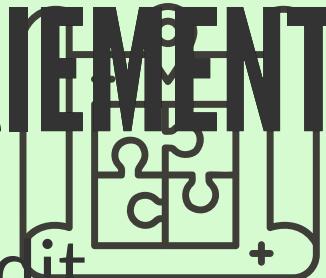


# EXTENSIONS AU UC1: TRAITER VENTE



- 7b. Payer par carte de crédit
  - ... information de crédit ... demande d'autorisation ... autorisation de paiement ...
- 7c. Payer par chèque
  - ...
  - chèque ... pièce d'identité ...
  - autorisation de paiement par chèque ...  
paiement par chèque ...

# SYSTÈMES D'AUTORISATION DE PAIEMENT



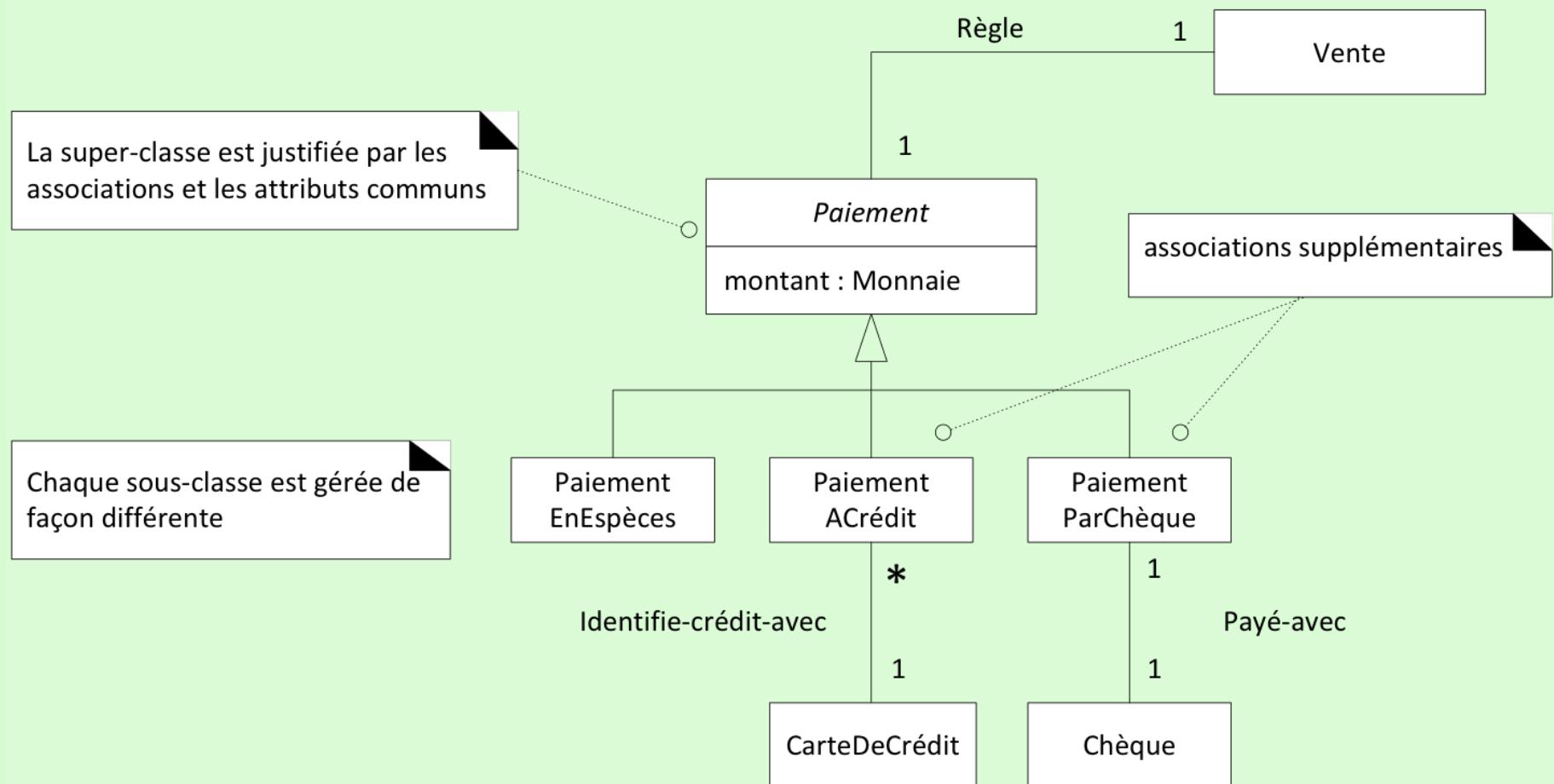
- RequêtePaiementACrédit et AccordCrédit
  - Abstractions des éléments dans l'activité d'autorisation de paiement
  - Pas nécessairement des informations (en termes de données transactionnelles) transmises sur le fil du réseau
  - **Classes conceptuelles toujours**



# EXEMPLE DANS POS



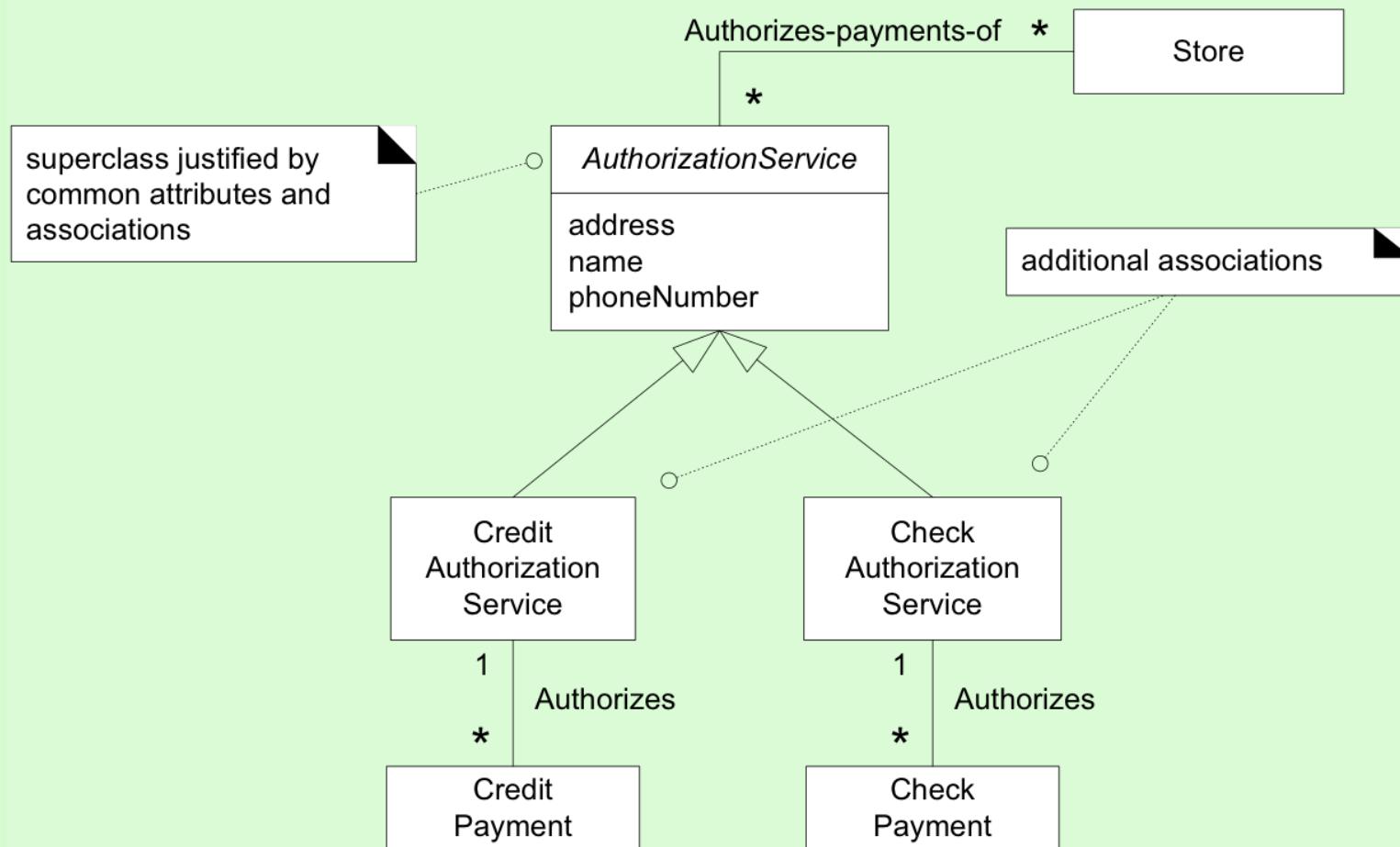
## Généralisation/spécialisation



# EXEMPLE DANS POS



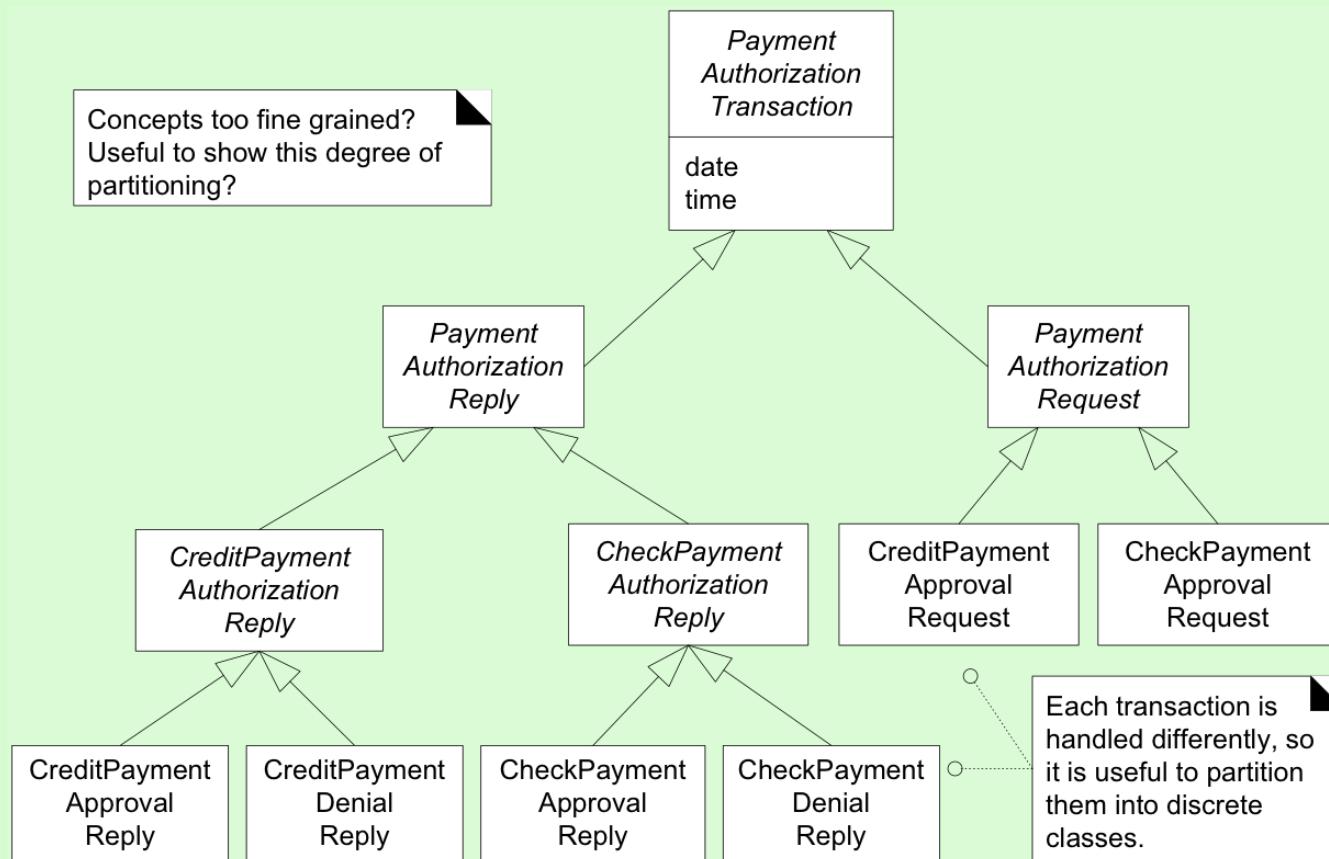
## Hiérarchie des services d'autorisation



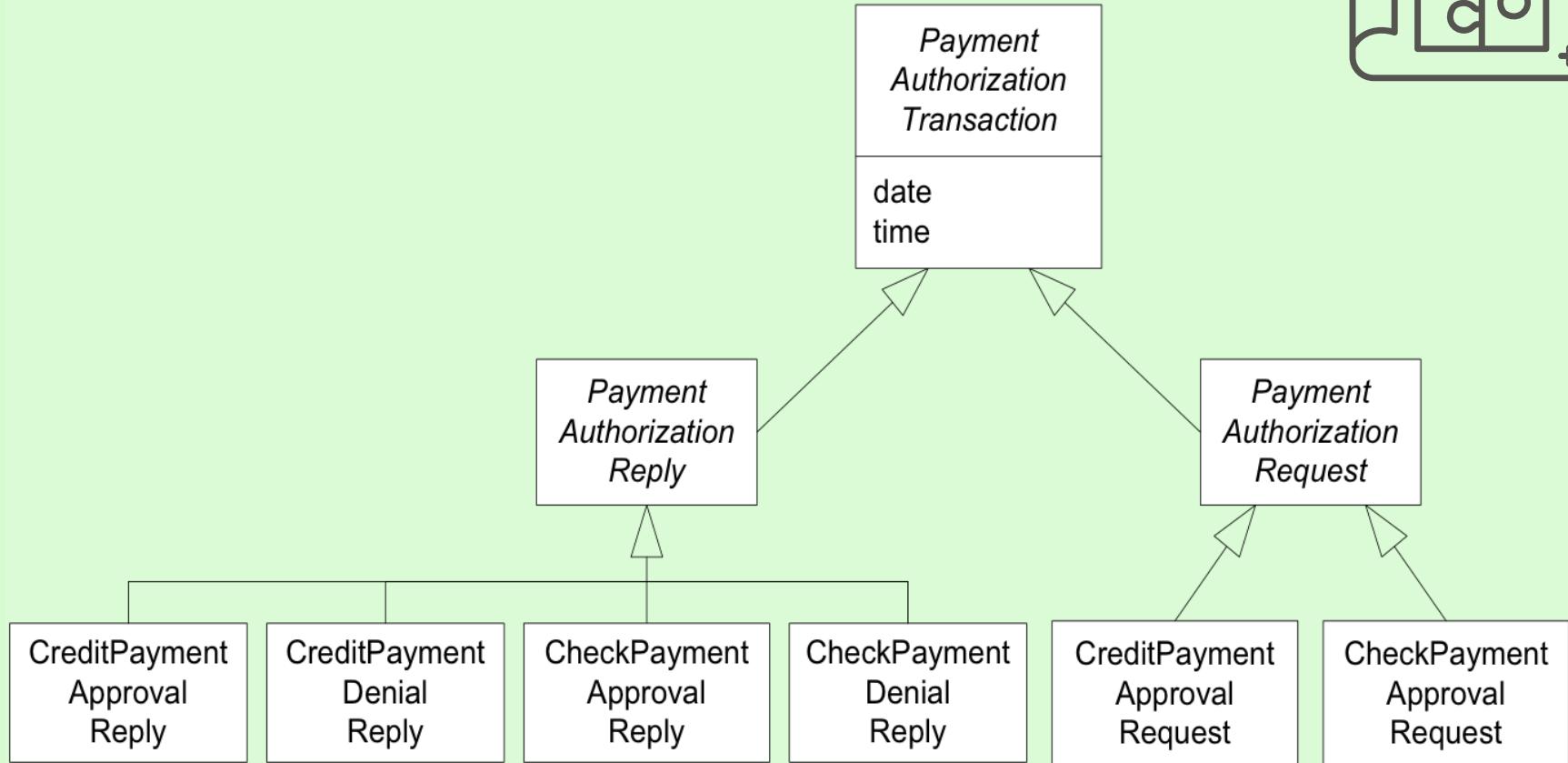
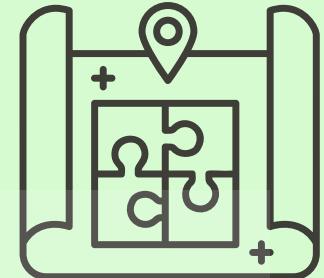
# TRANSACTIONS D'AUTORISATION



- Est-ce utile?



# TRANSACTIONS D'AUTORISATION



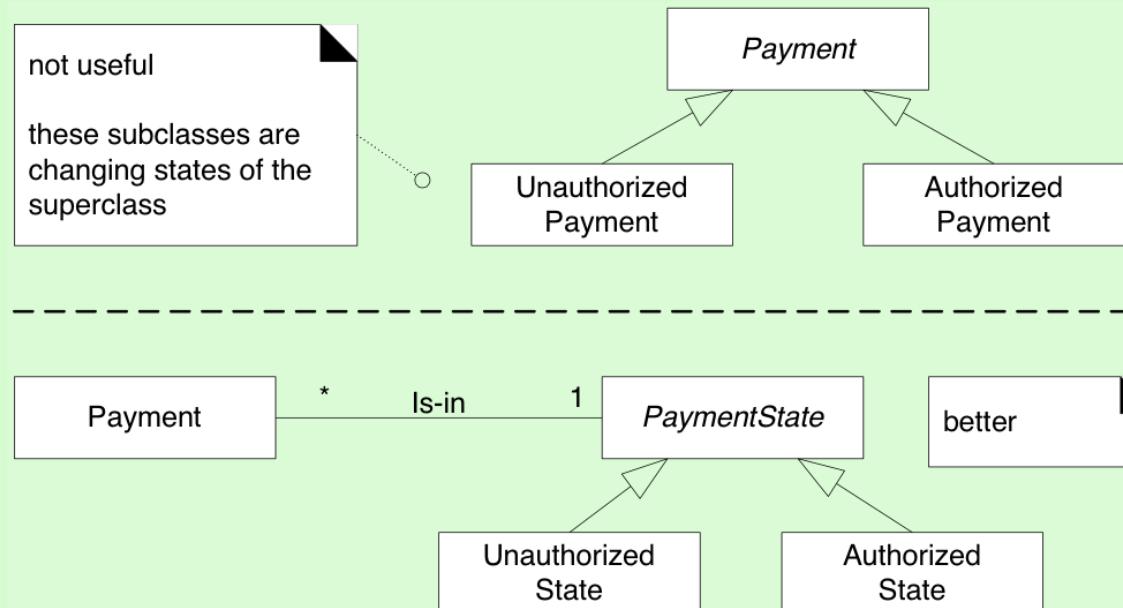
Le modèle du domaine n'est pas jugé en fonction de sa précision (justesse, correcte ou non) mais en terme de son utilité. Il n'est pas un but en soi



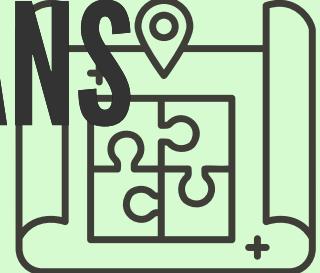
# MODÉLISATION DE CHANGEMENT D'ÉTAT



- Ne modélez pas les différents états possibles d'un concept comme des sous classes
- Deux solutions
  - définir une hiérarchie d'états et les associés à X ;
  - ignorer la représentation des états dans le MDD et faire un diagramme d'états

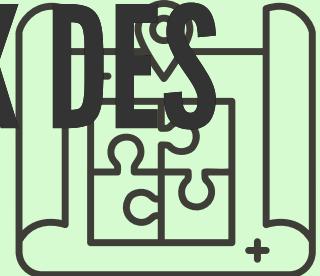


# HIERARCHIES ET HÉRITAGE DANS L'IMPLÉMENTATION

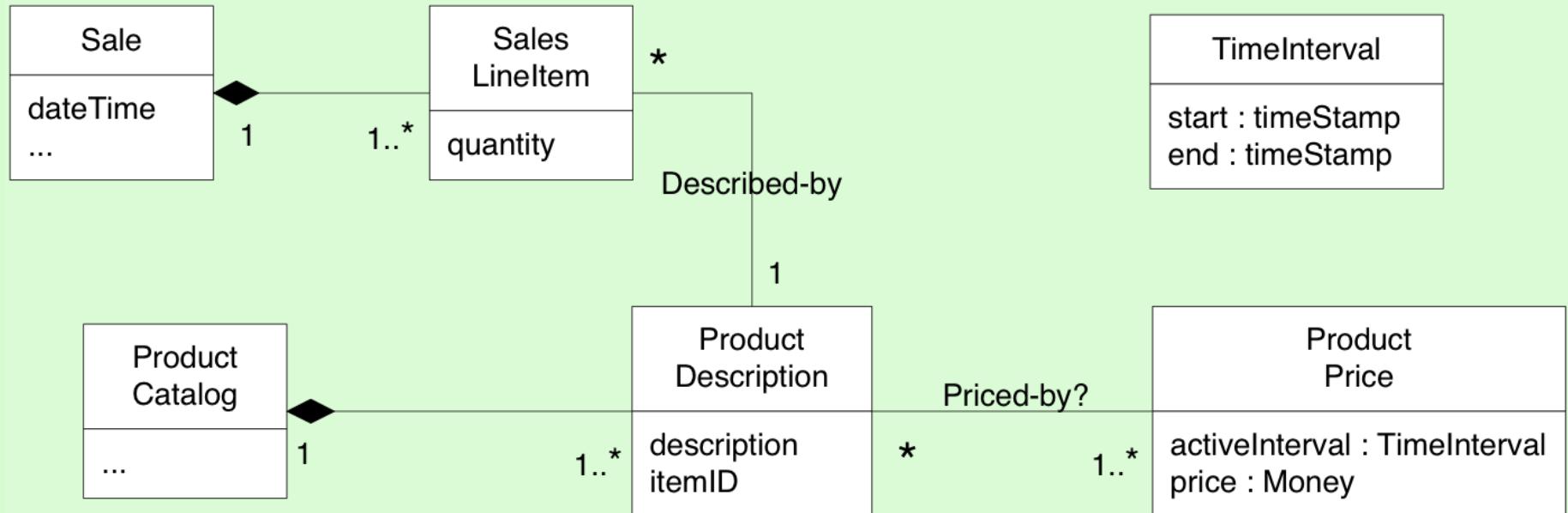


- Les hiérarchies de classes conceptuelles peuvent être reflétées ou non dans le modèle de conception
  - Par exemple, les classes paramétrées (templates) en C++/Java/C# permettent de réduire le nombre de classes dans l'implémentation
  - [www.tutorialspoint.com/cplusplus/cpp\\_templates](http://www.tutorialspoint.com/cplusplus/cpp_templates)

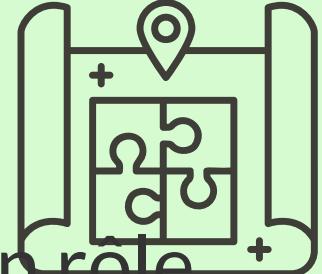
# INTERVALLES DE TEMPS ET PRIX DES PRODUITS



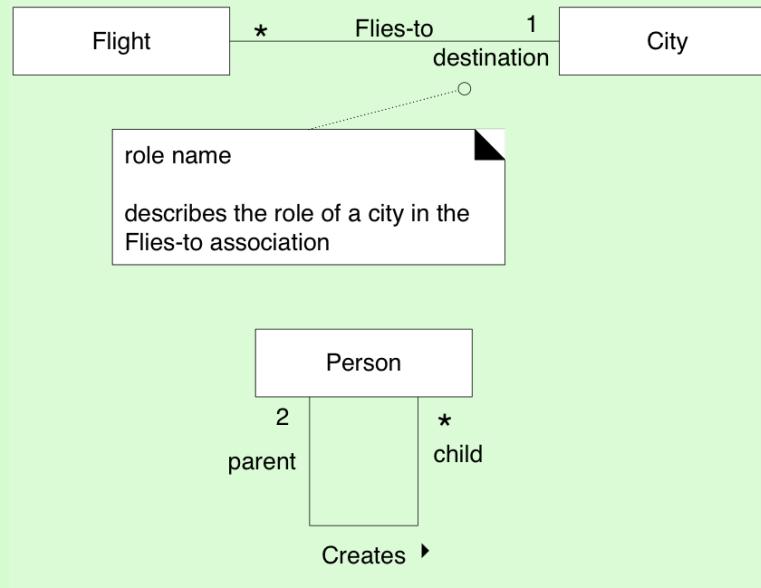
Correction d'une « erreur » de l'itération 1



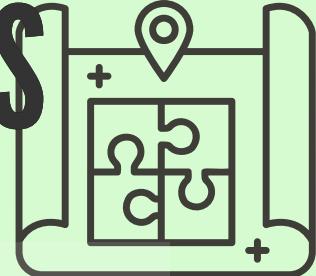
# NOM DE RÔLES



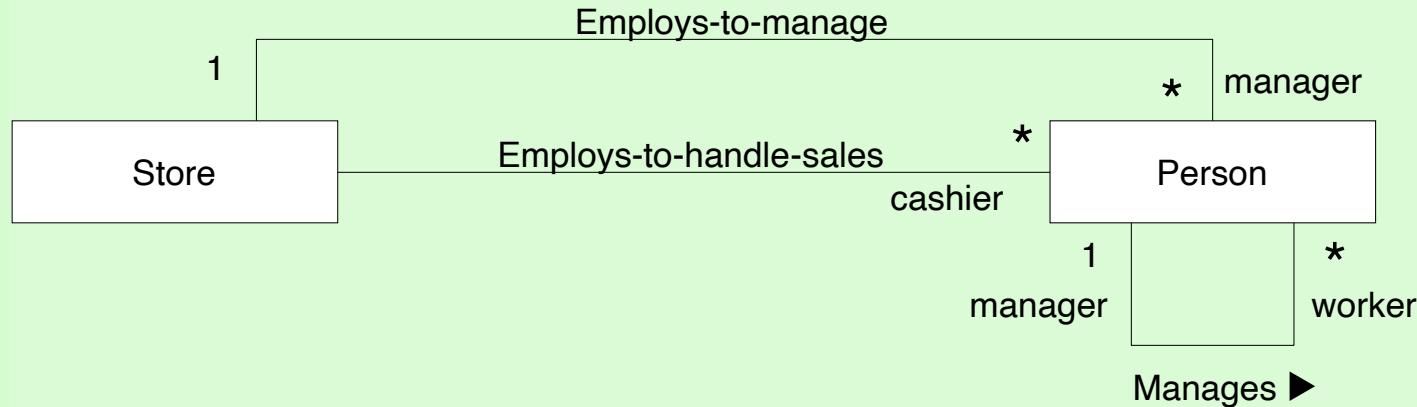
- Extrémité d'une association peut avoir un rôle
- Rôle possède plusieurs propriétés: nom et multiplicité



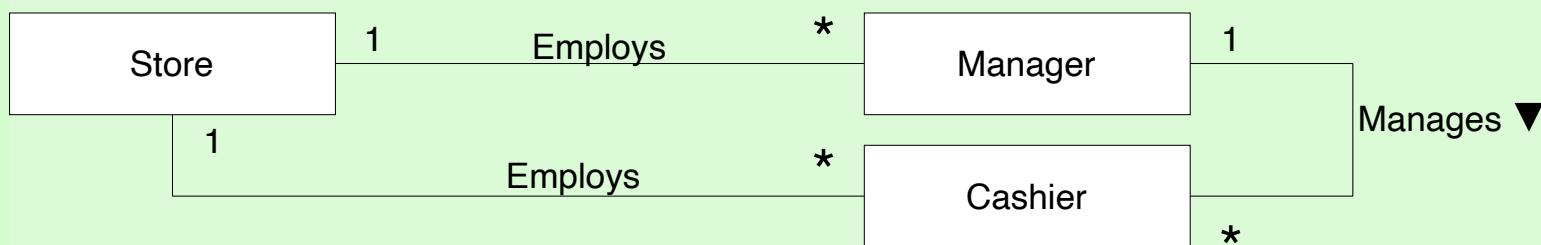
# CONCEPTS VS ASSOCIATIONS



roles in associations



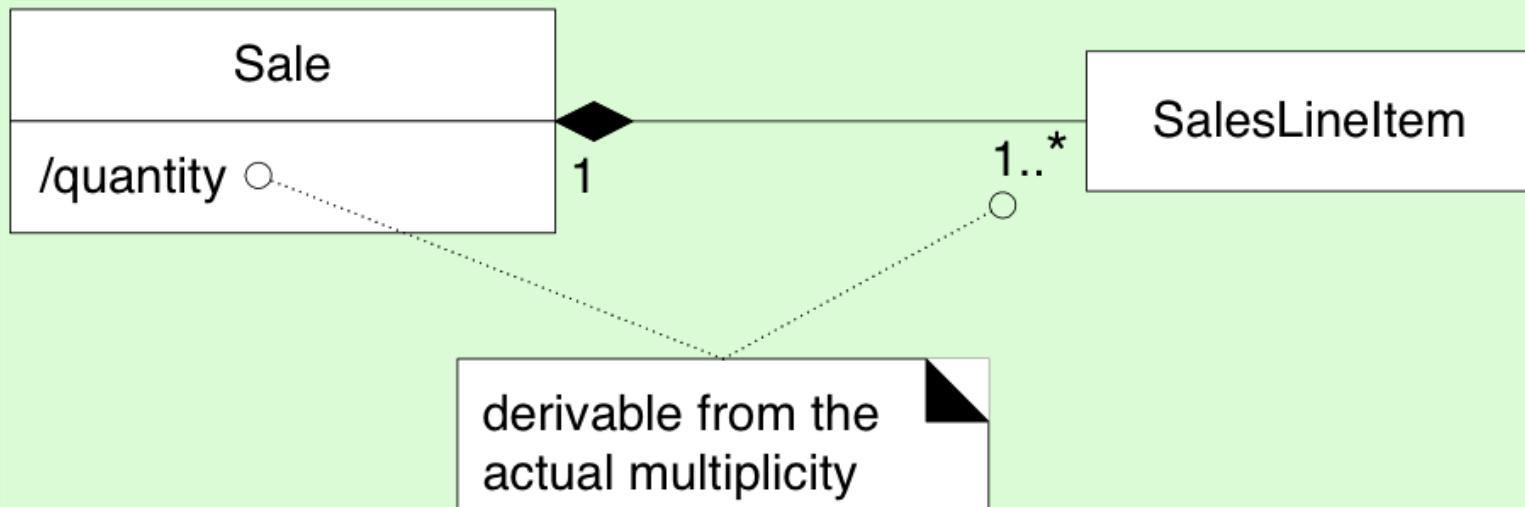
roles as concepts



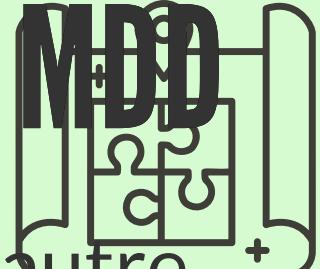
# ÉLÉMENTS DÉRIVÉS



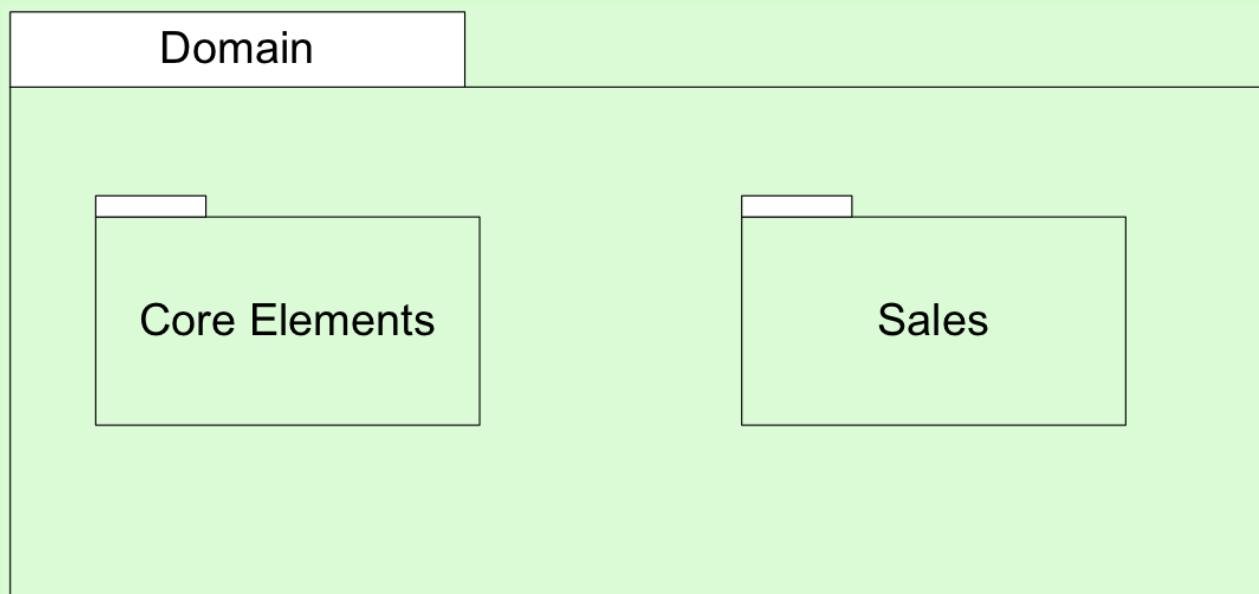
Évitez de faire apparaître les éléments dérivés dans les diagrammes,  
sauf si leur omission peut nuire à la compréhension



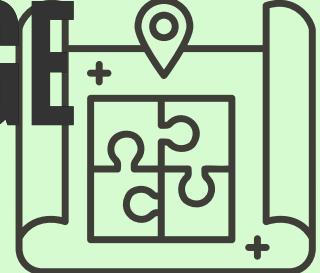
# PACKAGES POUR ORGANISER LE MDD



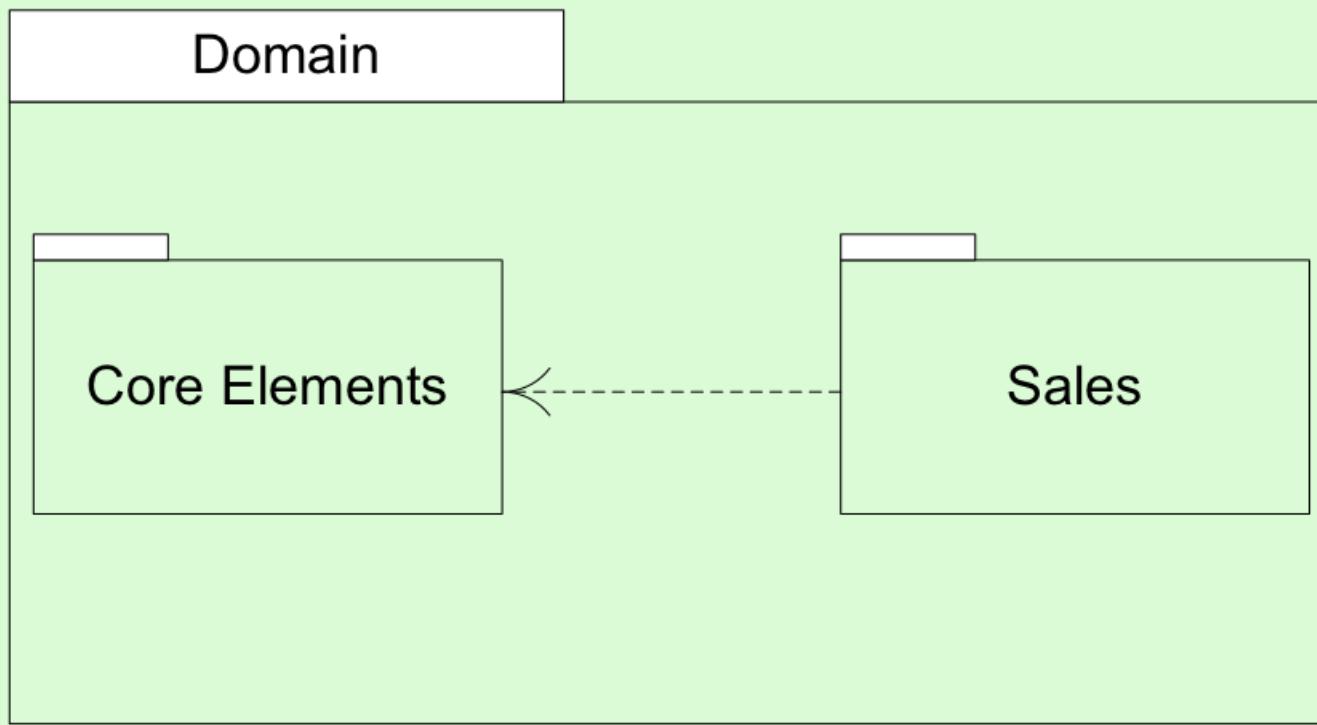
- Un élément peut être référencé dans un autre package
  - nom augmenté du nom du package



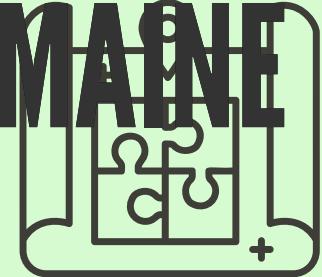
# DÉPENDANCE ENTRE PACKAGE



- Package qui dépend d'un autre
  - indicateur de couplage
  - entre certains éléments du premier et du deuxième

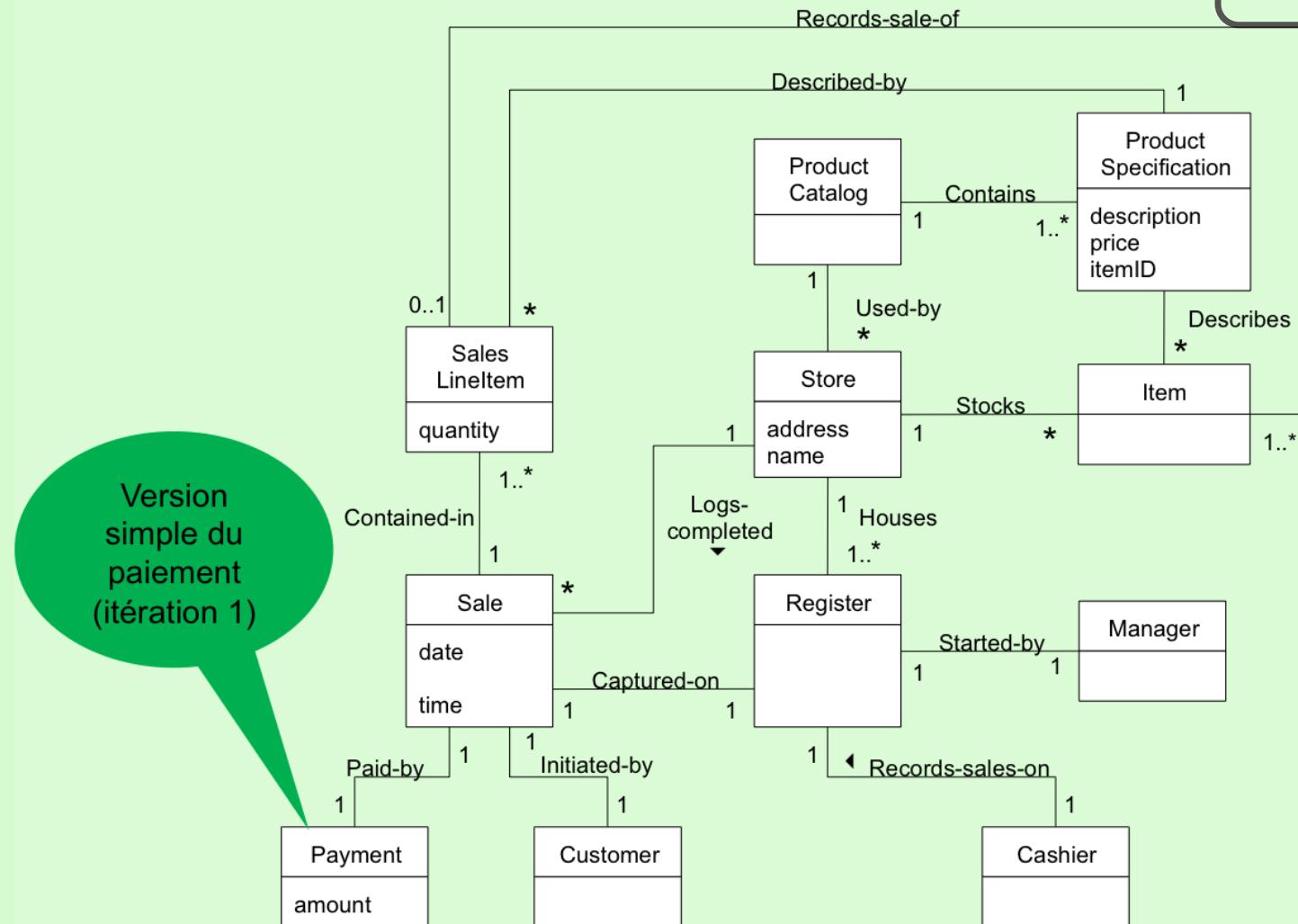


# PARTITIONNER LE MODÈLE DU DOMAINE

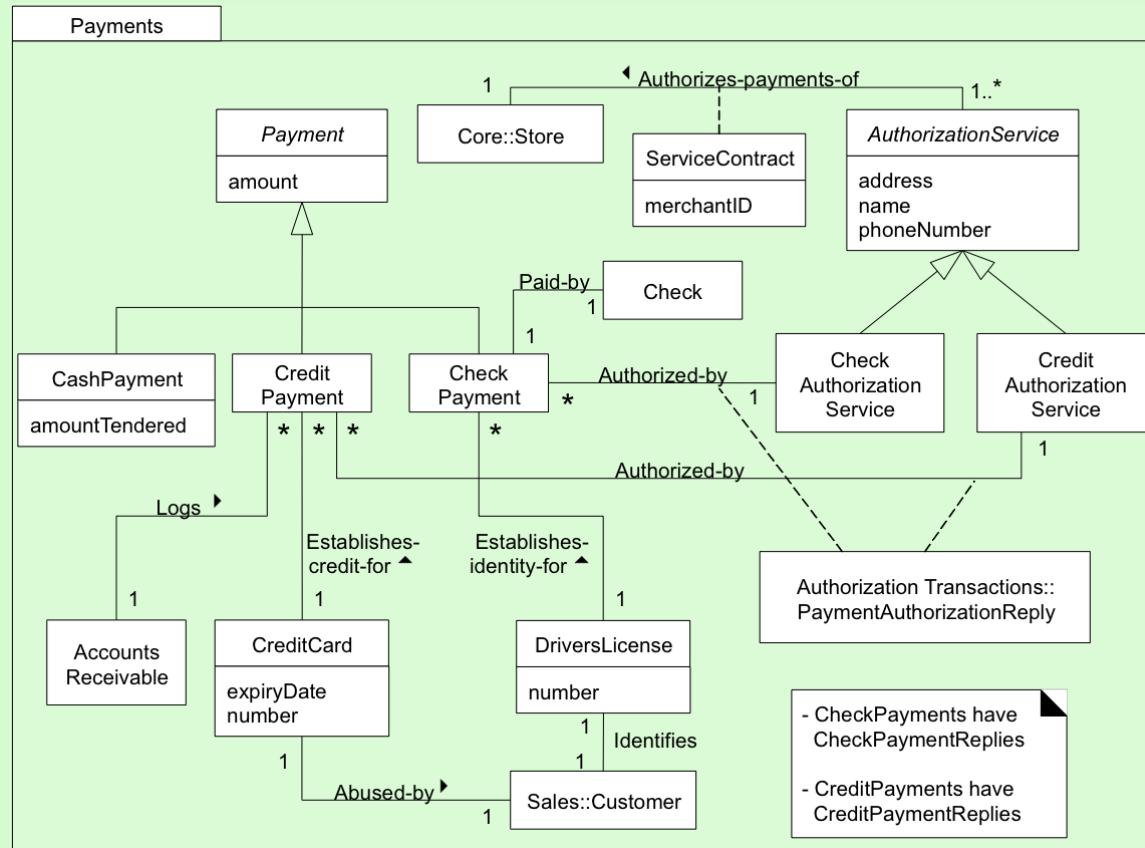


- Placer en package les éléments
  - Cohésion relationnelle
    - qui sont fortement associés
    - situés dans la même hiérarchie de classes
  - Cohésion fonctionnelle
    - situés dans le même domaine
    - participant au même cas d'utilisation

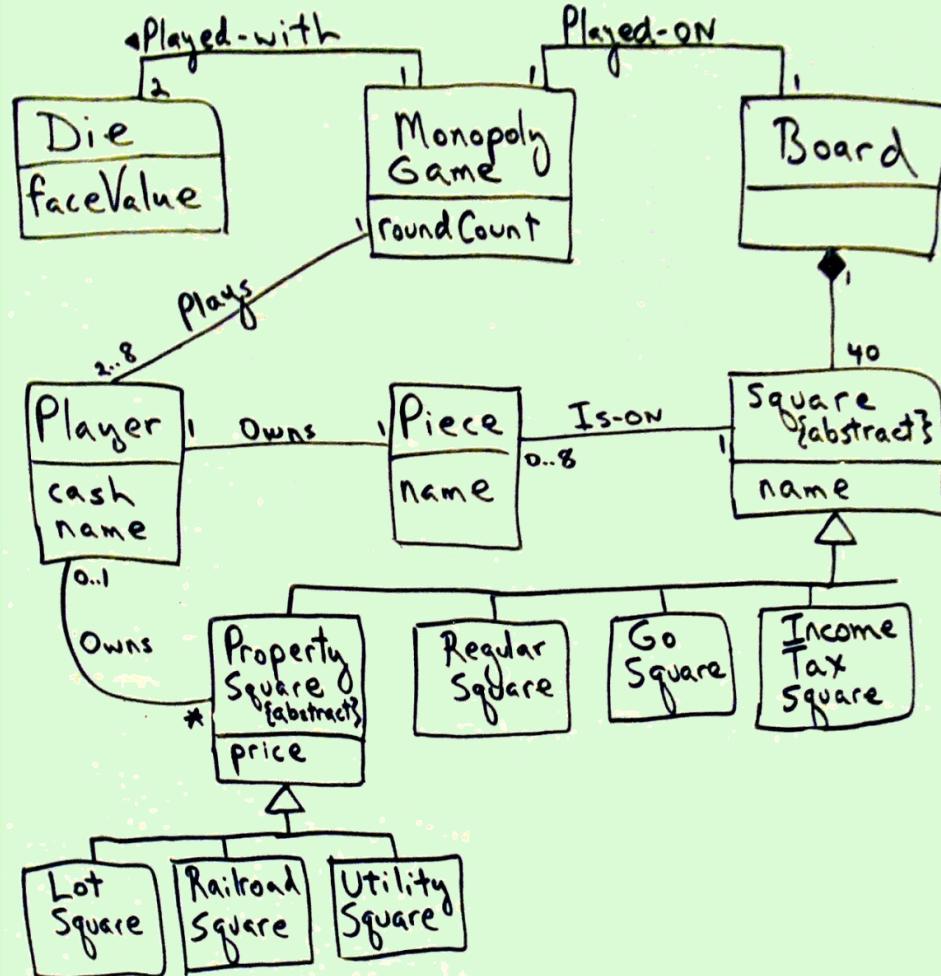
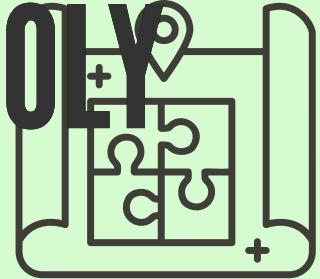
# MODÈLE DU DOMAINE (1)



# MODÈLE DU DOMAINE (2)

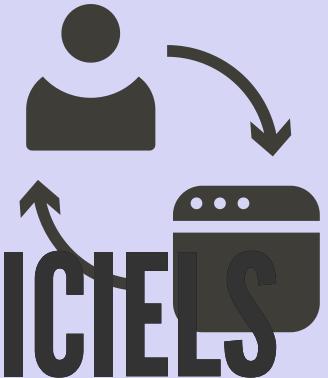


# AFFINEMENTS DU MDD MONOPOLY



# LOG210 SÉANCE #10

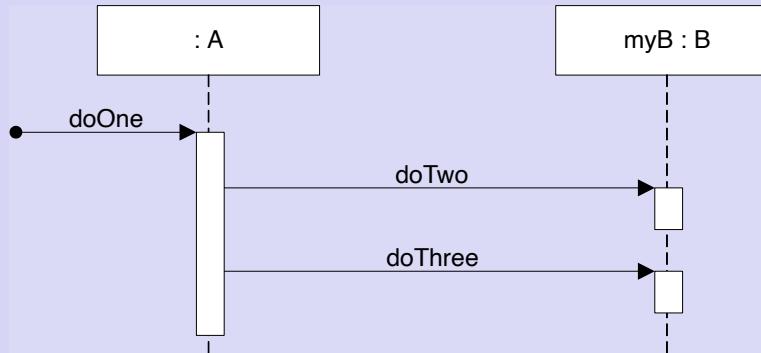
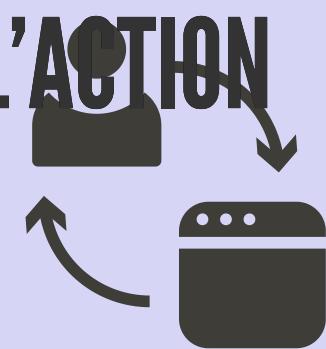
## ANALYSE ET CONCEPTION DE LOGICIELS



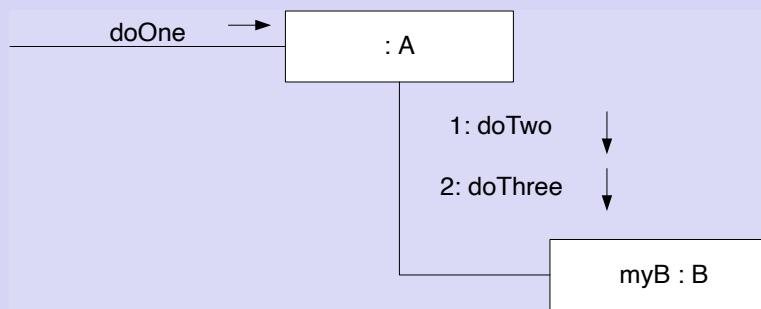
1. Affinement du MDD
2. Diagramme d'interaction ← 9.33m
3. Diagramme d'état
4. Diagramme d'activité

# DIAGRAMMES D'INTERACTION MODÉLISENT L'ACTION

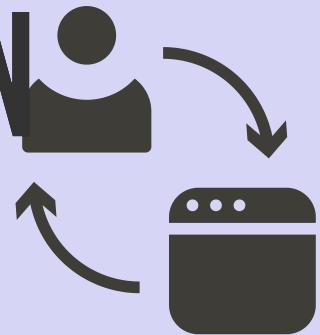
## Diagrammes de séquence



## Diagrammes de communication

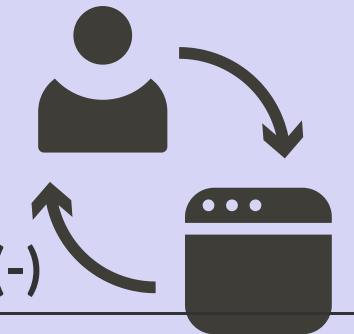


# DIAGRAMME D'INTERACTION



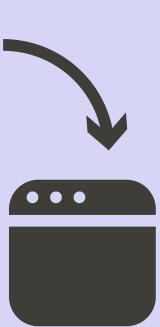
- Pour détailler les opérations
- Annoté avec les GRASP

# COMPARAISON



Type	Forces (+)	Faiblesses (-)
Séquence	Indique clairement la séquence et l'ordonnancement des messages. Grande richesse de la notation	Ajout de nouveaux objets doit être vers la droite: consomme l'espace horizontal
Communication	Économique en termes d'espace, permet d'ajouter des objets dans les deux dimensions	Rend plus difficile la lecture des séquences de messages. Moins d'options de notation

# RAPPEL DE LOG121



## CLASSE VS. INSTANCE

Vente

Classe

:Vente

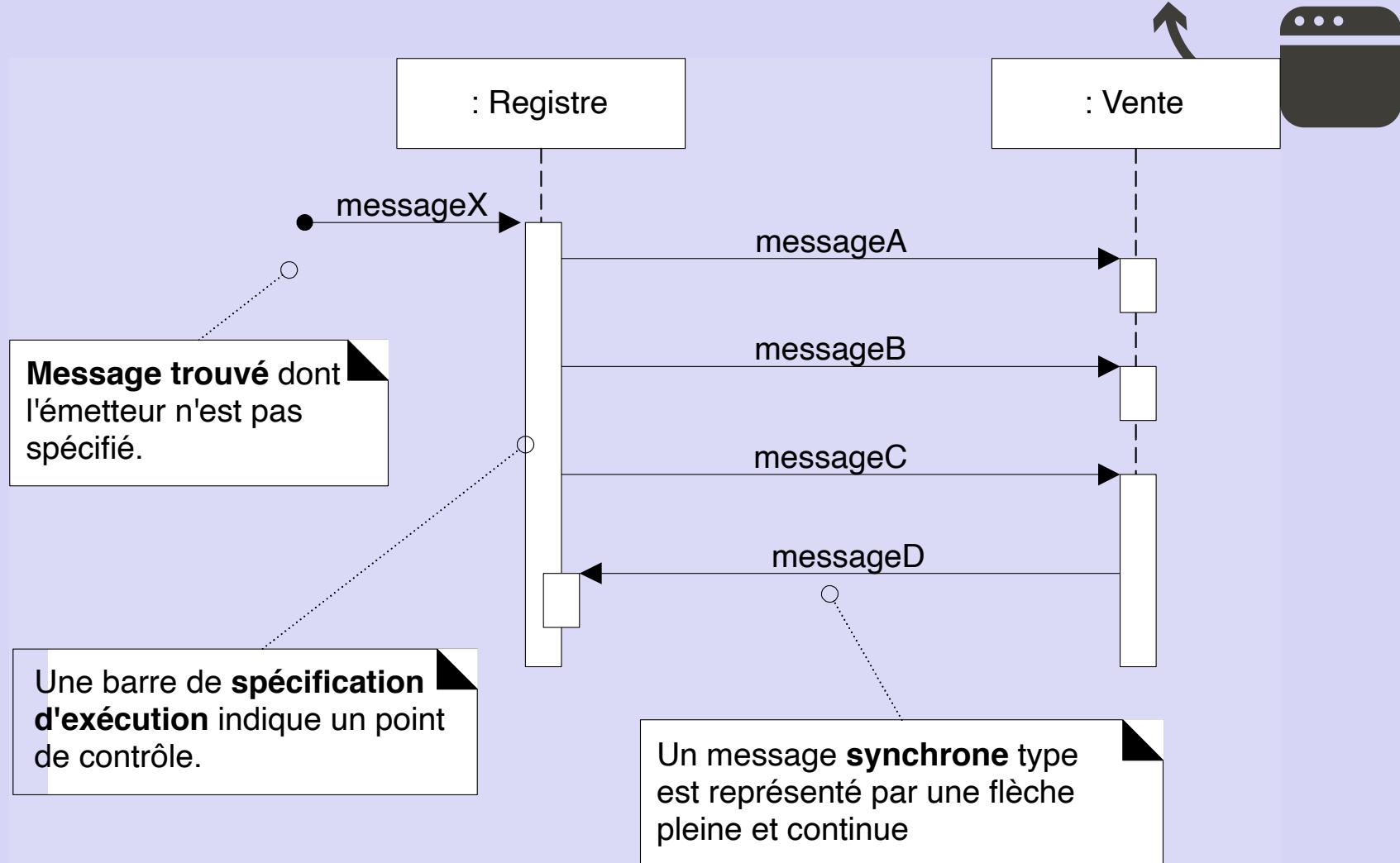
Instance

v:Vente

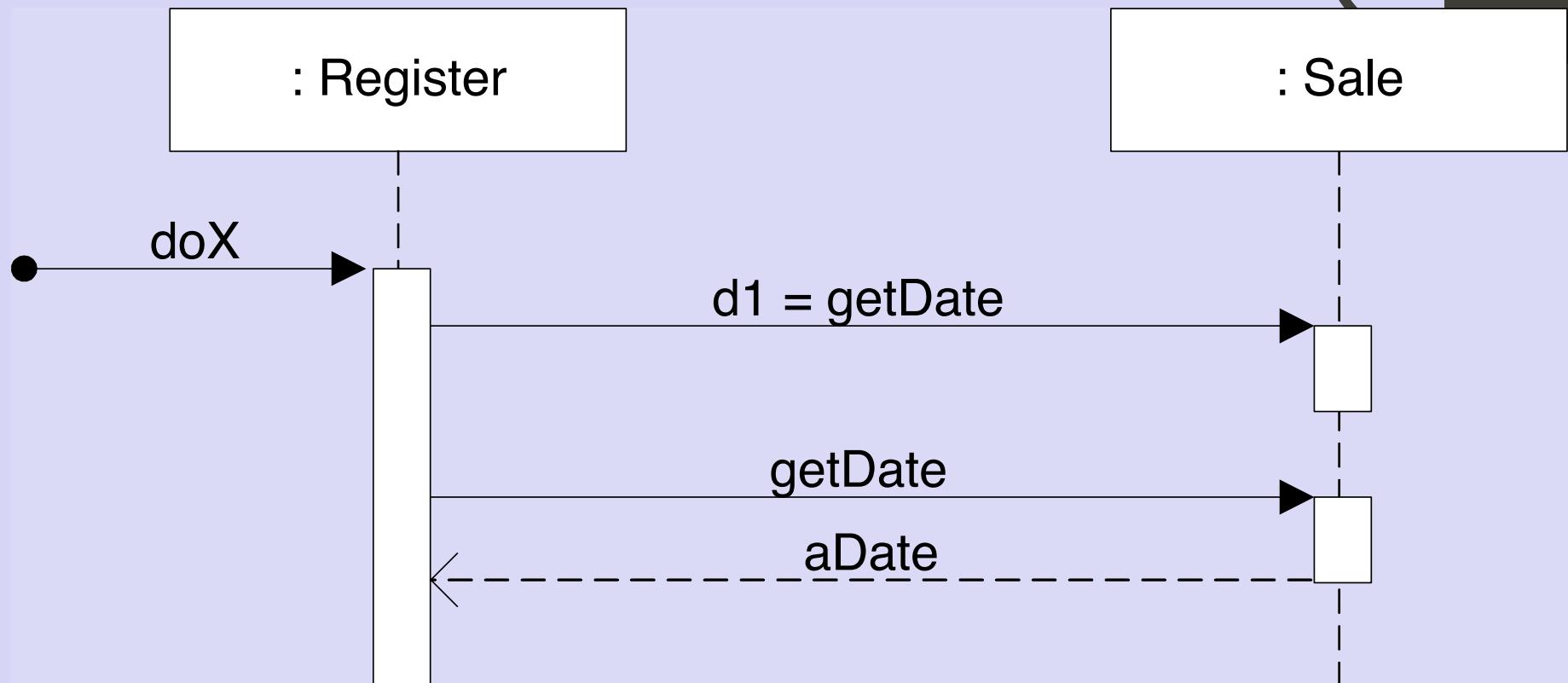
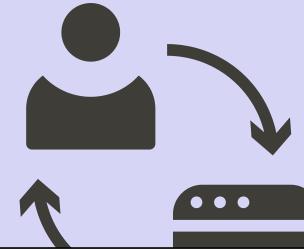
Instance  
nommée



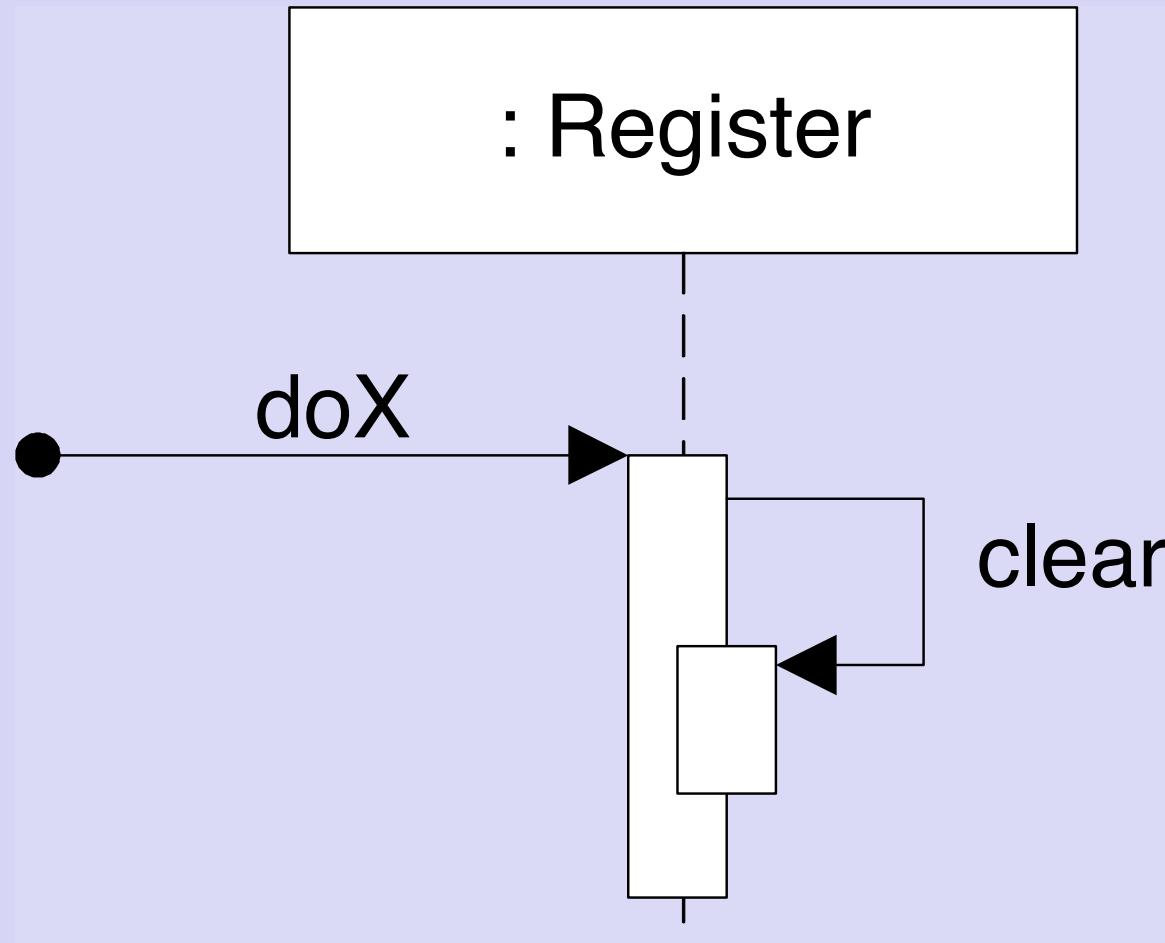
# NOTATION: DIAGRAMMES DE SÉQUENCE



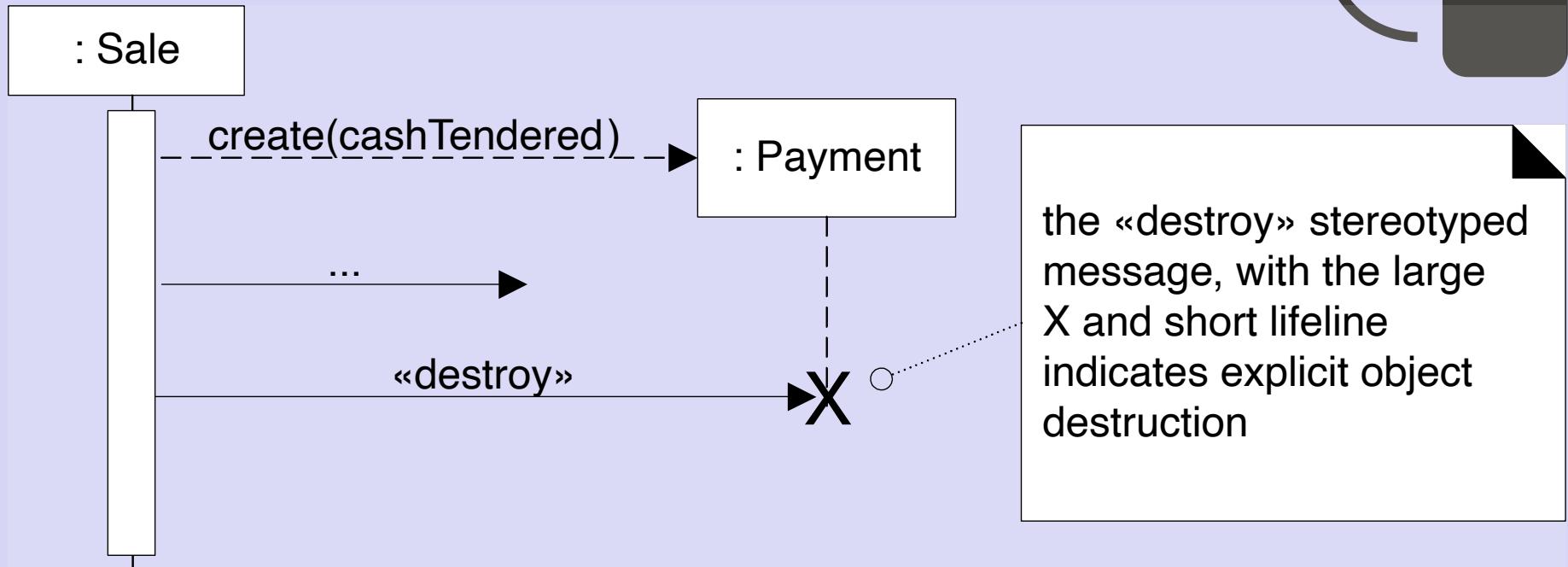
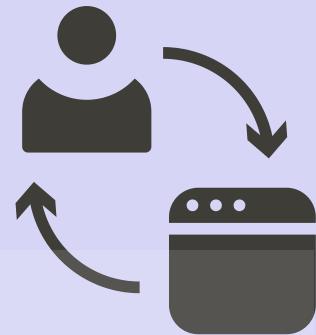
# RÉSULTAT D'UN MESSAGE



# MESSAGE D'UN OBJET À LUI-MÊME



# DESTRUCTION D'UN OBJET



# ITÉRATION SUR UNE COLLECTION (NOTATION EXPLICITE)

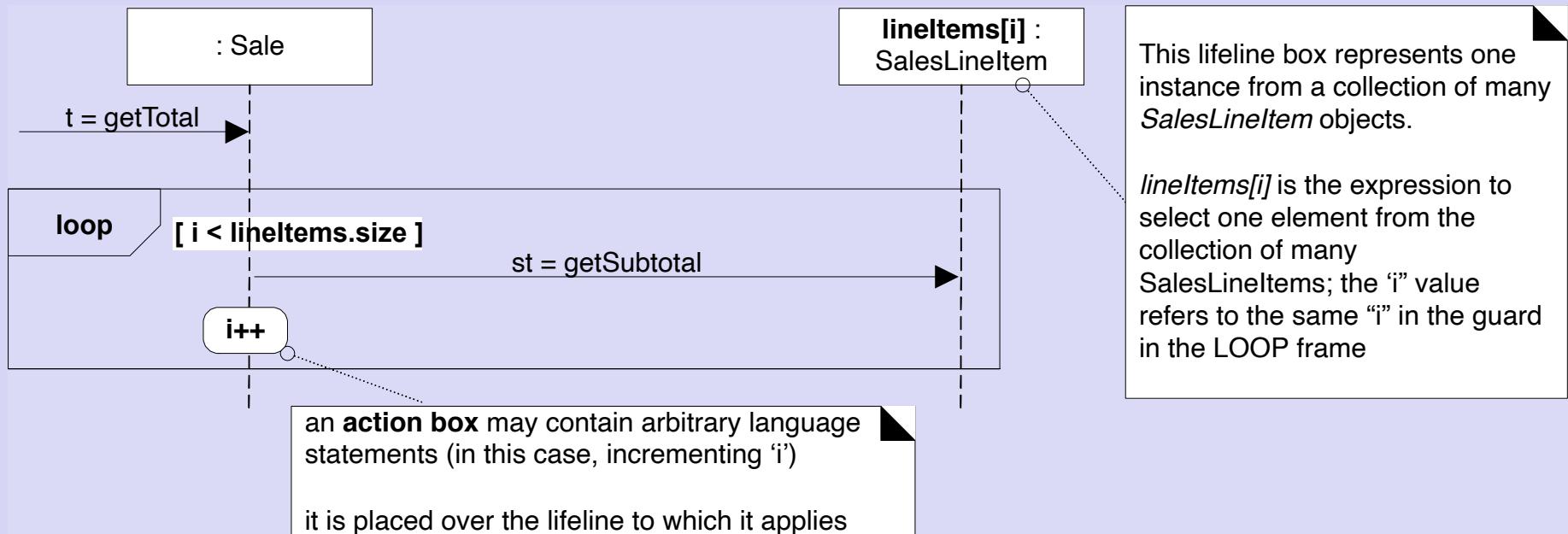
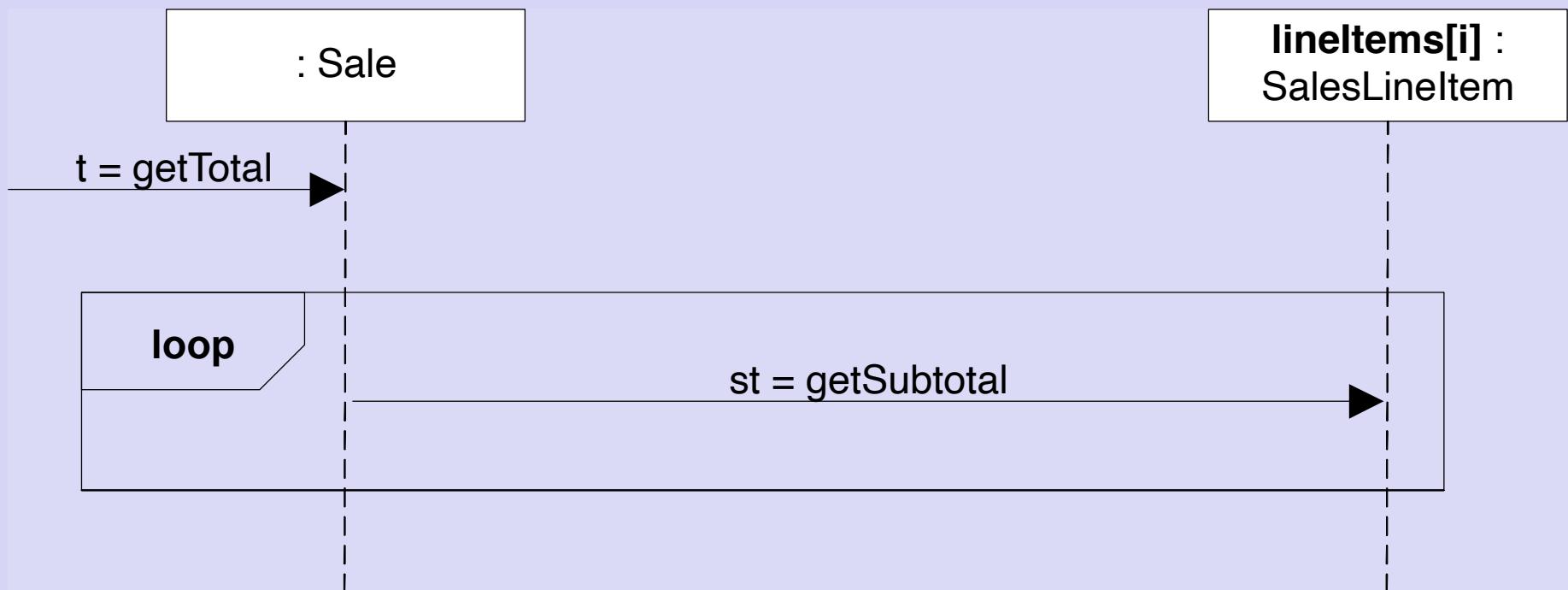


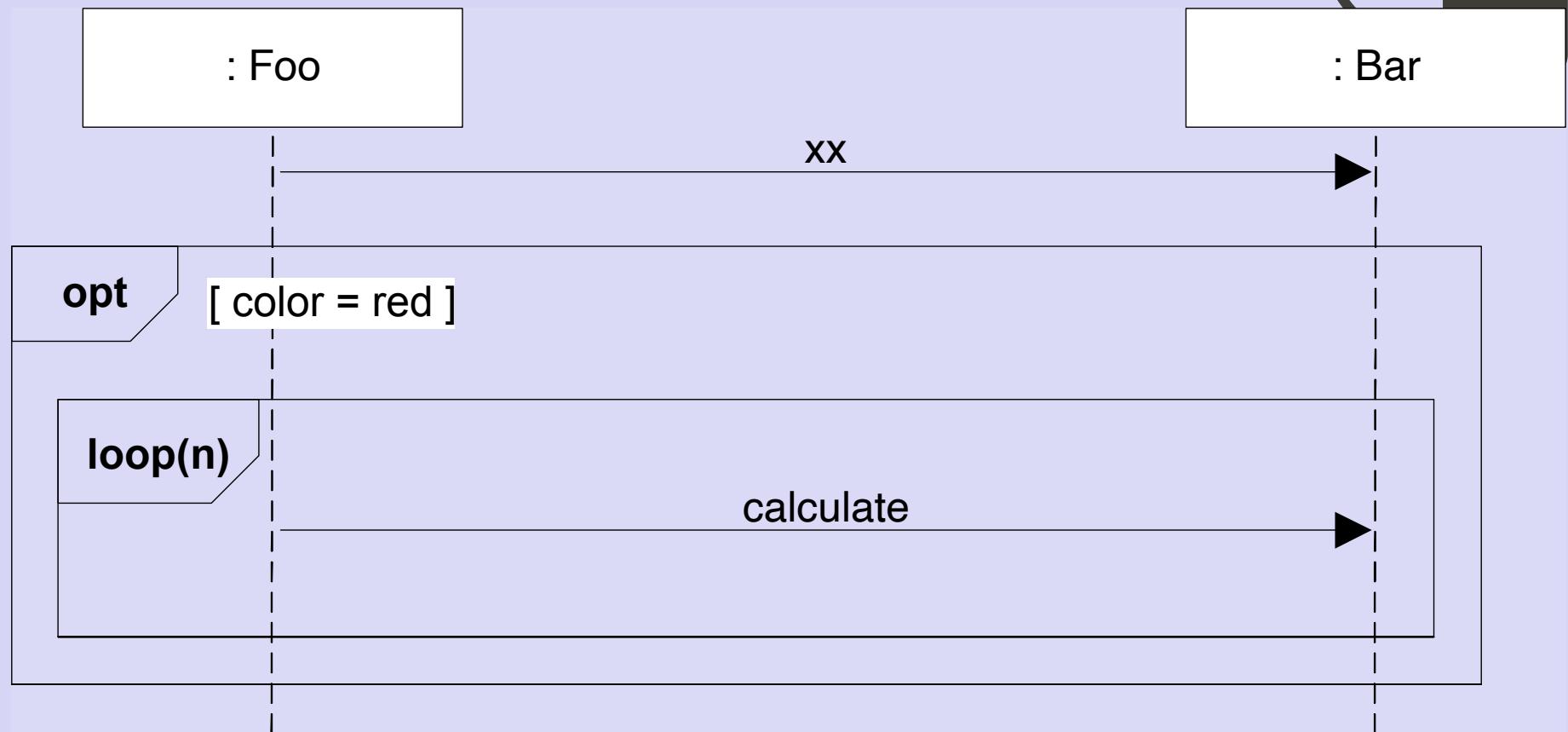
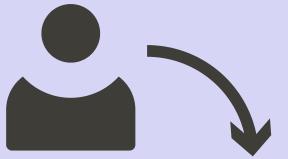
fig F14.16, A15.16



# ITÉRATION SUR UNE COLLECTION (NOTATION IMPLICITE)



# IMBRICATION DE CADRES



# APPELS ASYNCHRONES ET OBJETS ACTIFS

a stick arrow in UML implies an asynchronous call

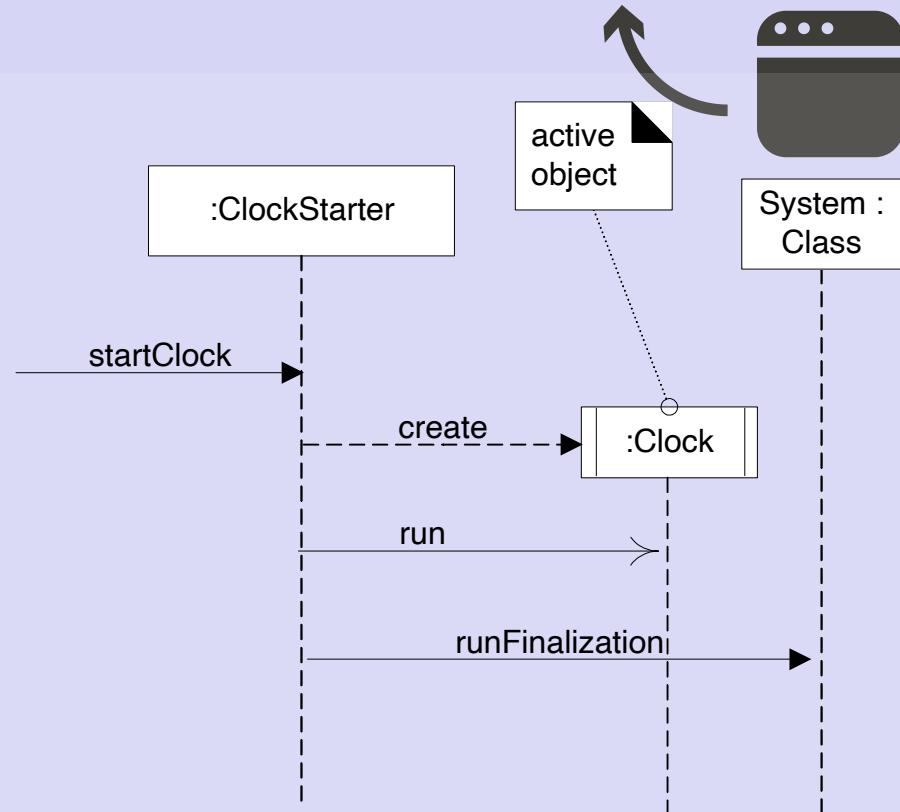
a filled arrow is the more common synchronous call

In Java, for example, an asynchronous call may occur as follows:

```
// Clock implements the Runnable interface
Thread t = new Thread( new Clock() );
t.start();
```

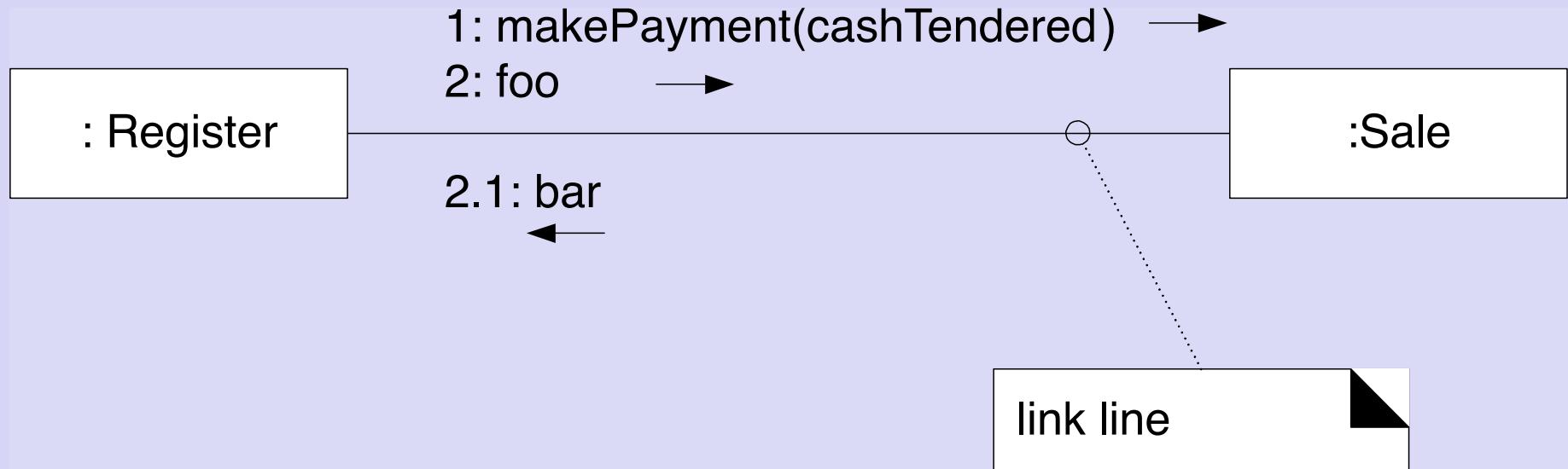
the asynchronous *start* call always invokes the *run* method on the *Runnable* (*Clock*) object

to simplify the UML diagram, the *Thread* object and the *start* message may be avoided (they are standard “overhead”); instead, the essential detail of the *Clock* creation and the *run* message imply the asynchronous call



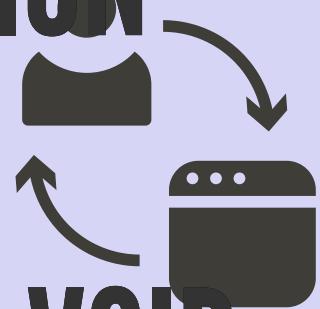
# DIAGRAMME DE COMMUNICATION

- Liens et messages



# DIAGRAMME DE COMMUNICATION

- Liens et messages

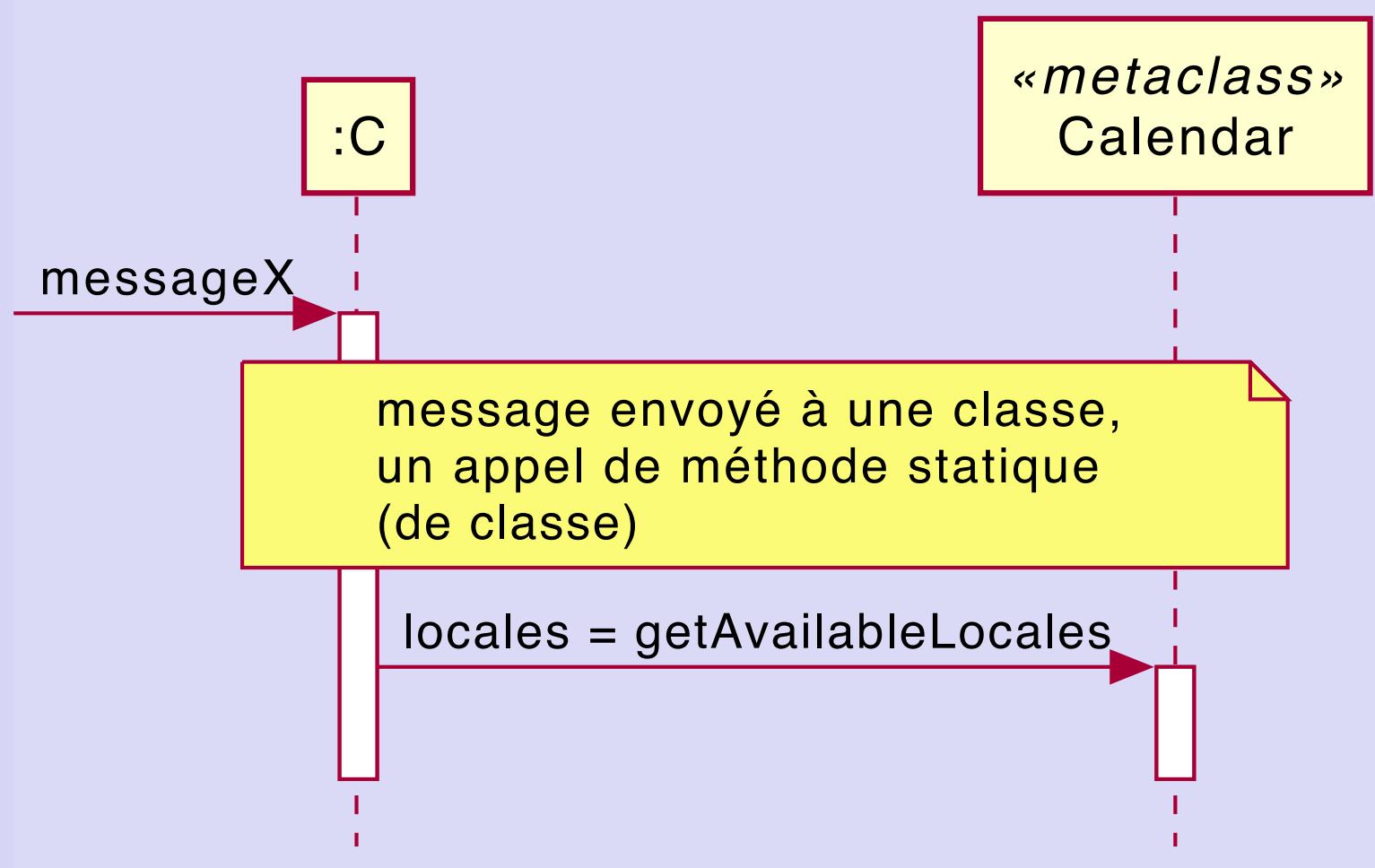


LARMAN/F14.24, A15.24 NO TE: VOIR

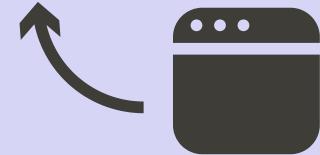
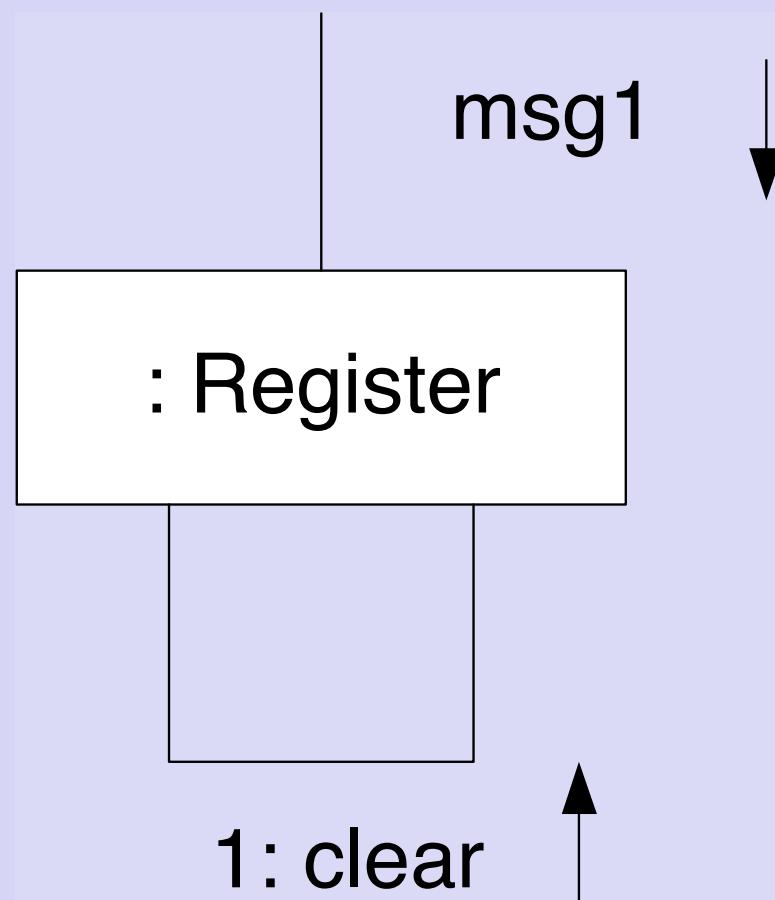
SOLUTION DANS LA SECTION EXERCICE  
PLUS BAS.

MESSAGE POUR UNE CLASSE

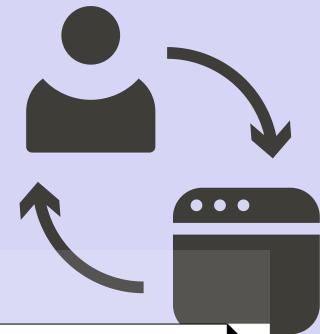




# MESSAGE D'UN OBJET À LUI-MÊME

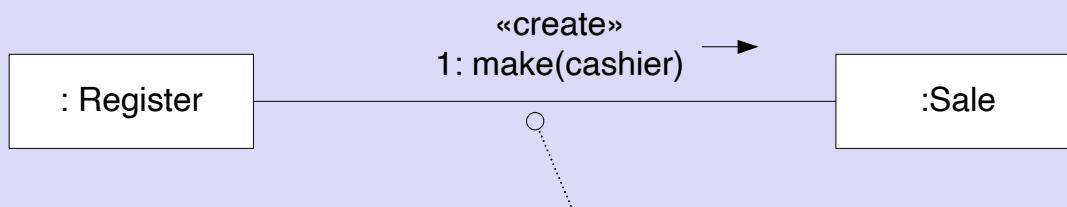
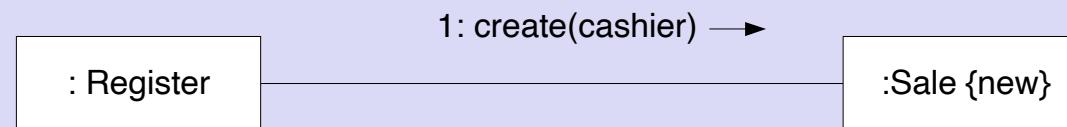
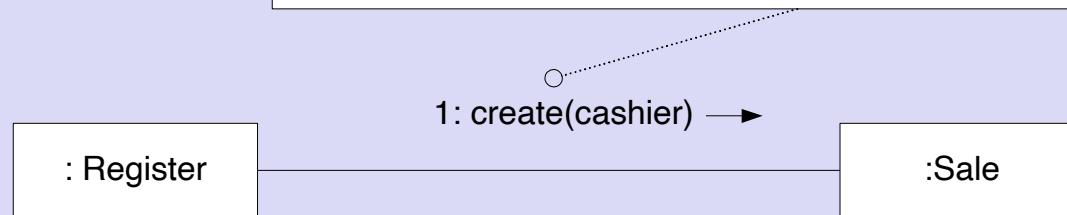


# CRÉATION D'INSTANCE



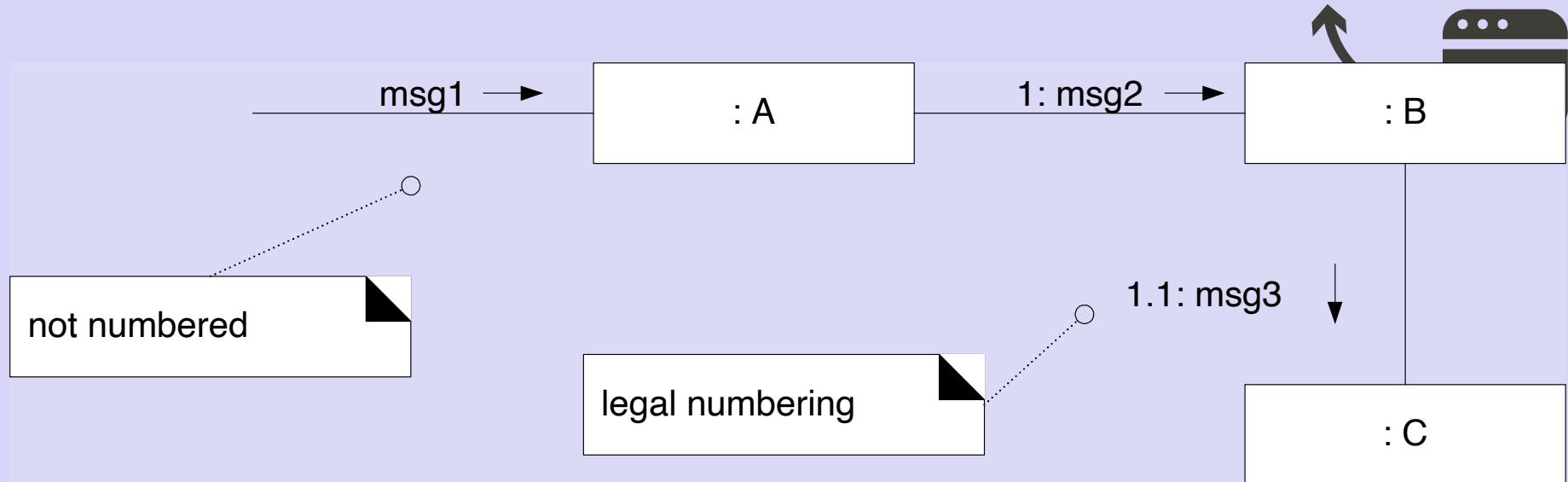
Three ways to show creation in a communication diagram

create message, with optional initializing parameters. This will normally be interpreted as a constructor call.

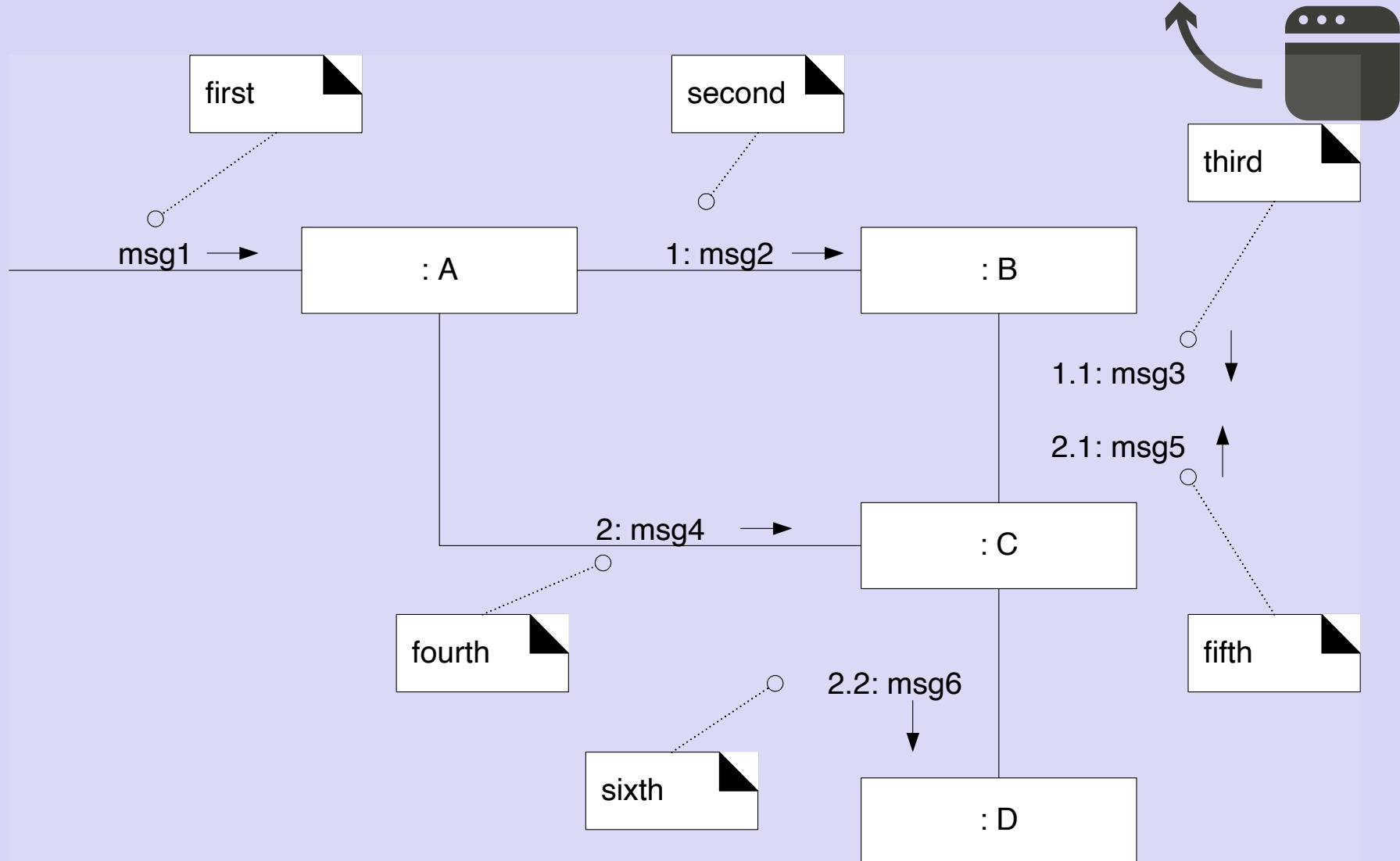


if an unobvious creation message name is used, the message may be stereotyped for clarity

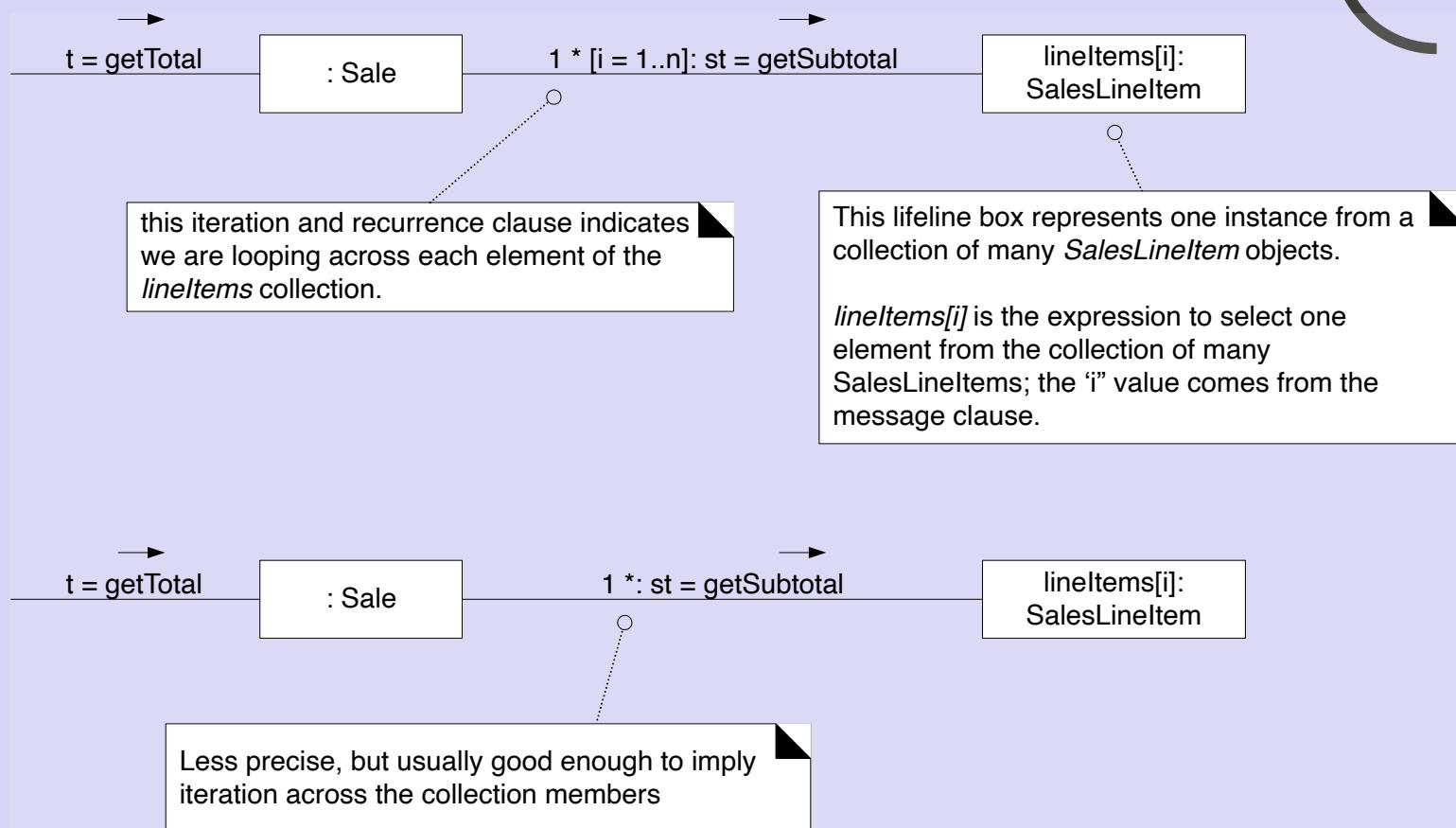
# AFFECTATION DE NUMÉROS D'ORDRE



# AFFECTATION DE NUMÉROS D'ORDRE

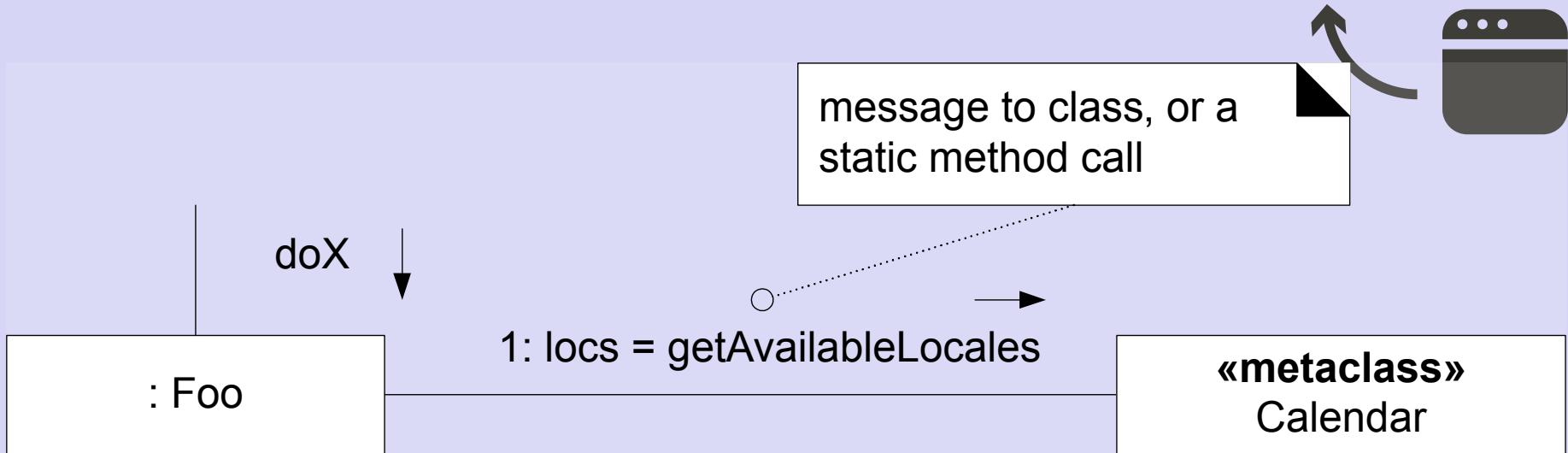


# ITÉRATION SUR UNE COLLECTION

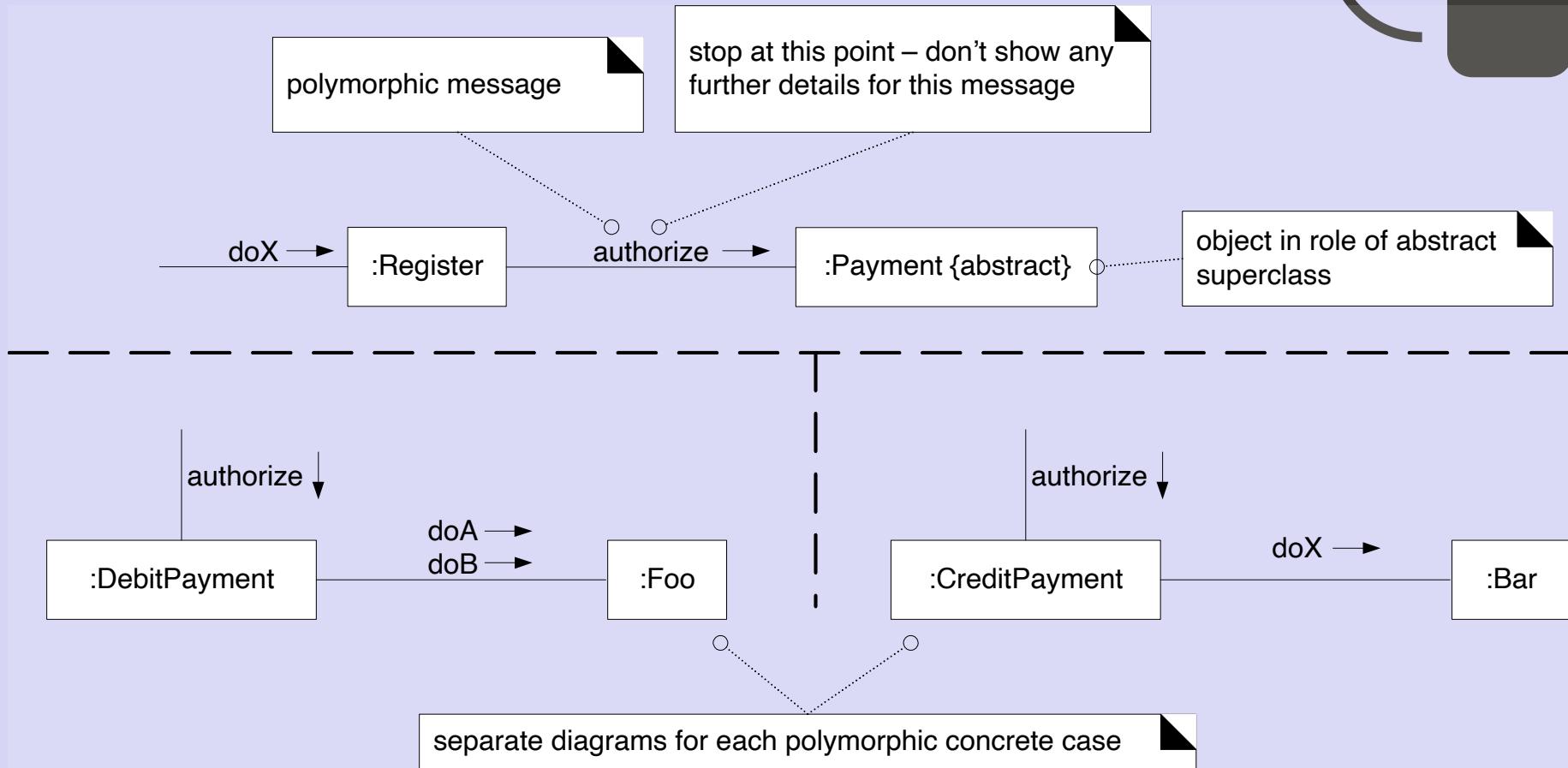
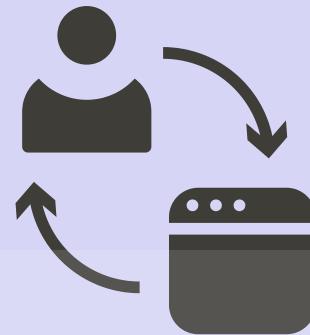


## collection en java

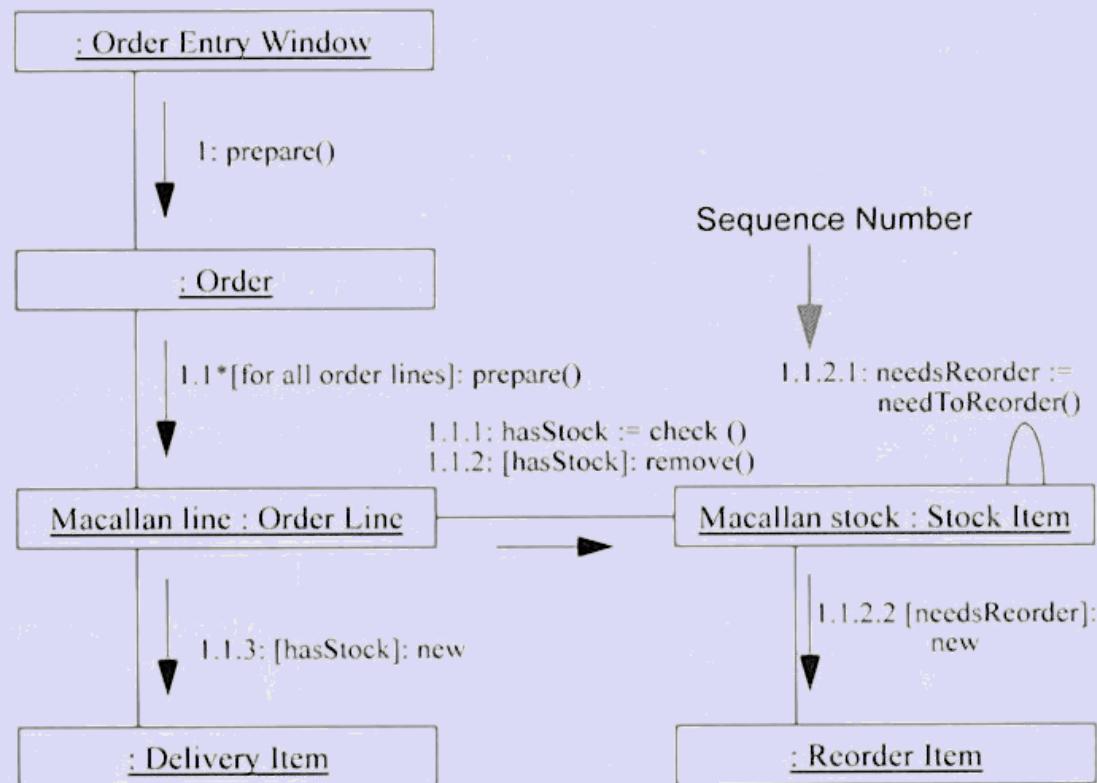
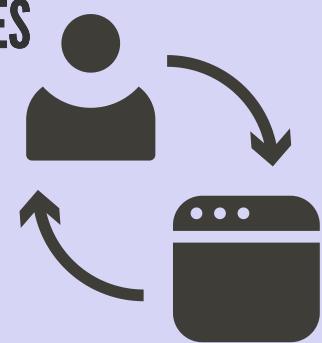
# MESSAGE VERS UNE CLASSE



# MESSAGES POLYMORPHES



# \*EXERCICE - TRANSFORMER EN DIAGRAMME DE SÉQUENCE, DE CLASSES

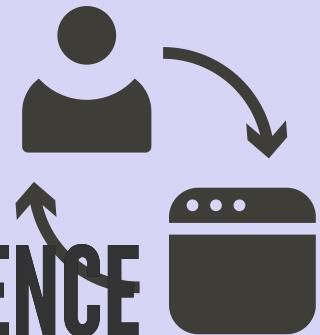


**Figure 5-5: Collaboration Diagram with Decimal Numbering**

ref: UML Distilled, 2nd ed., Fowler, Scott, Addison Wesley, 2000.

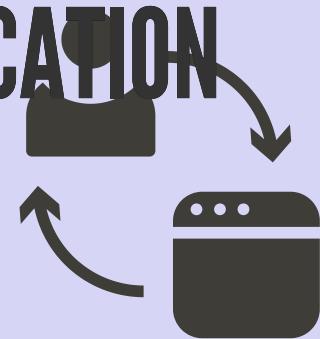
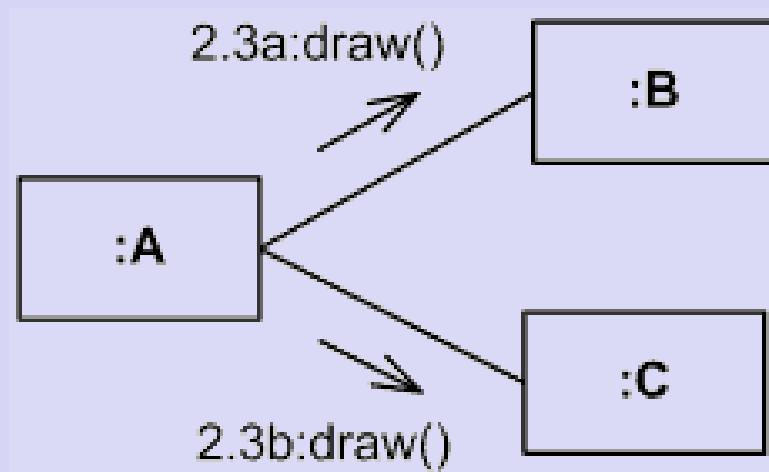
# \*EXERCICE

## TRANSFORMER EN DIAGRAMME DE SÉQUENCE



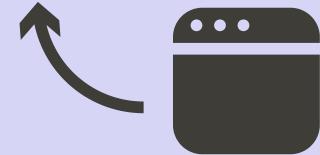
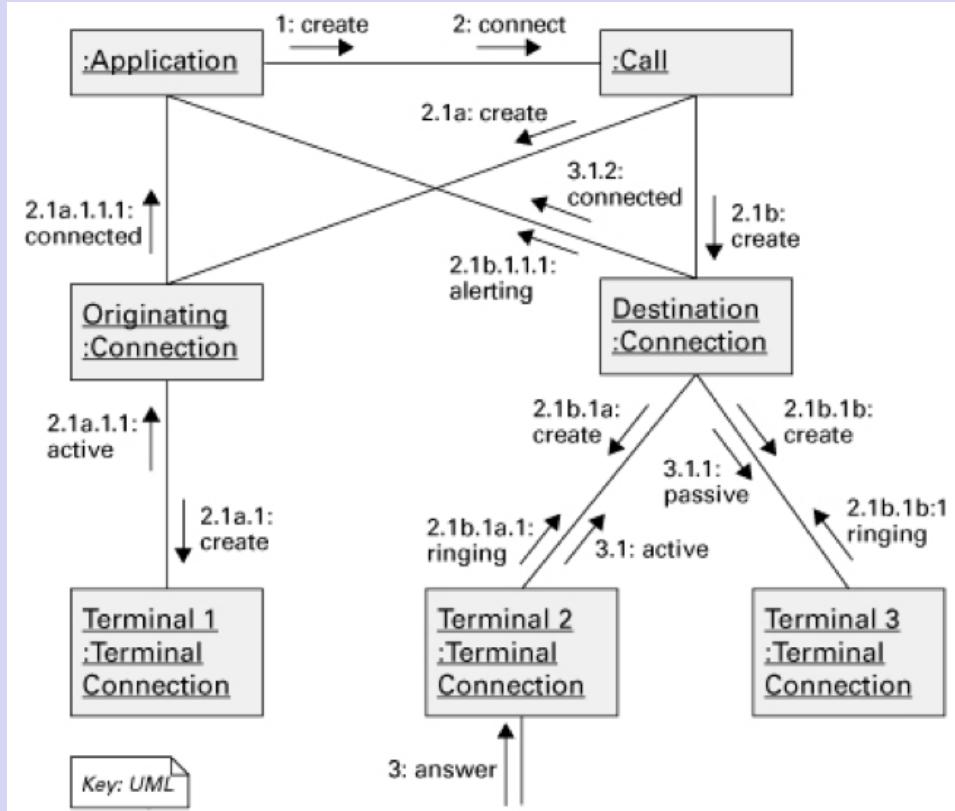
larman/F14.24, A15.24

# PARALLÉLISME: DIAGRAMME DE COMMUNICATION



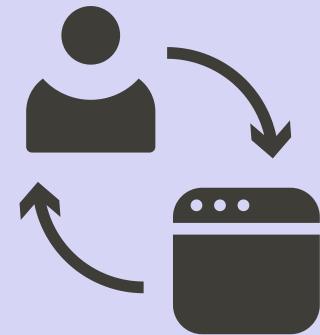
- For example,
  - messages 2.3a and 2.3b are concurrent within activation 2.3,
  - Name represents a concurrent thread of control.
  - Instance of A sends draw () messages concurrently to instance of B and to instance of C
- Réf: <https://www.uml-diagrams.org/communication-diagrams.html>

# DIAGRAMME DE COLLABORATION UML

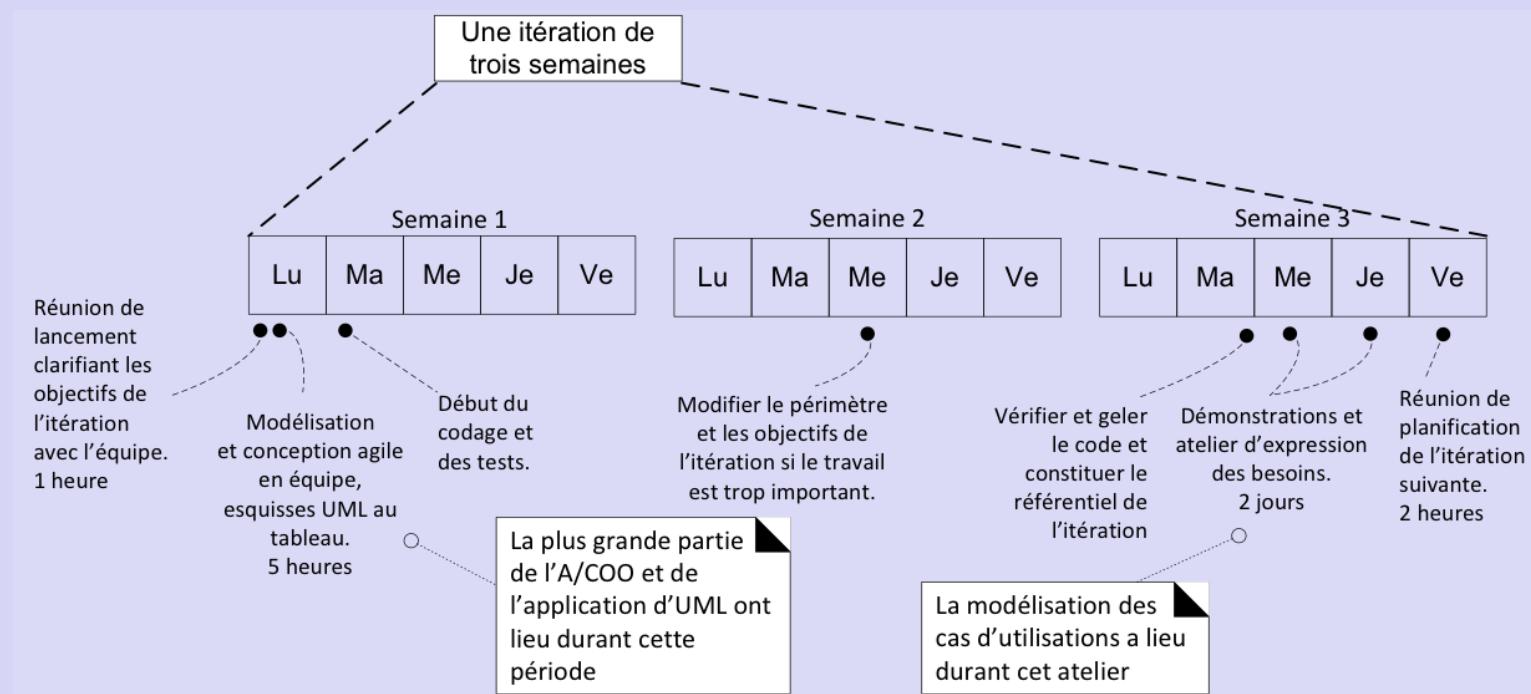


Réf: Documenting Software Architectures: Views and Beyond, 2e édition, Felix Bachmann, Len Bass, Paul C. Clements, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith A. Stafford

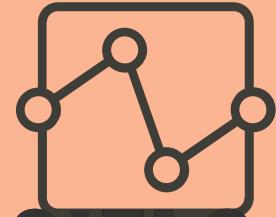
# RÉSUMÉ



- Deux types de diagrammes d'interaction
  - de séquence et de communication
- Agilité est importante dans la modélisation
  - Habiliter de faire rapidement des modèles en UML



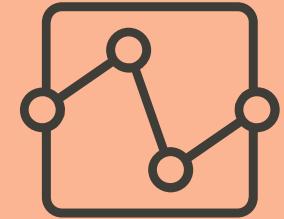
# LOG210 SÉANCE #10



## ANALYSE ET CONCEPTION DE LOGICIELS

1. Affinement du MDD
2. Diagramme d'interaction
3. Diagramme d'état ← 26.34m
4. Diagramme d'activité

# DIAGRAMME D'ÉTAT

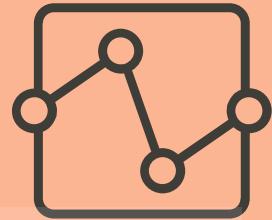


Pour modéliser les comportements.

Created by Rudez Studio  
from the Noun Project

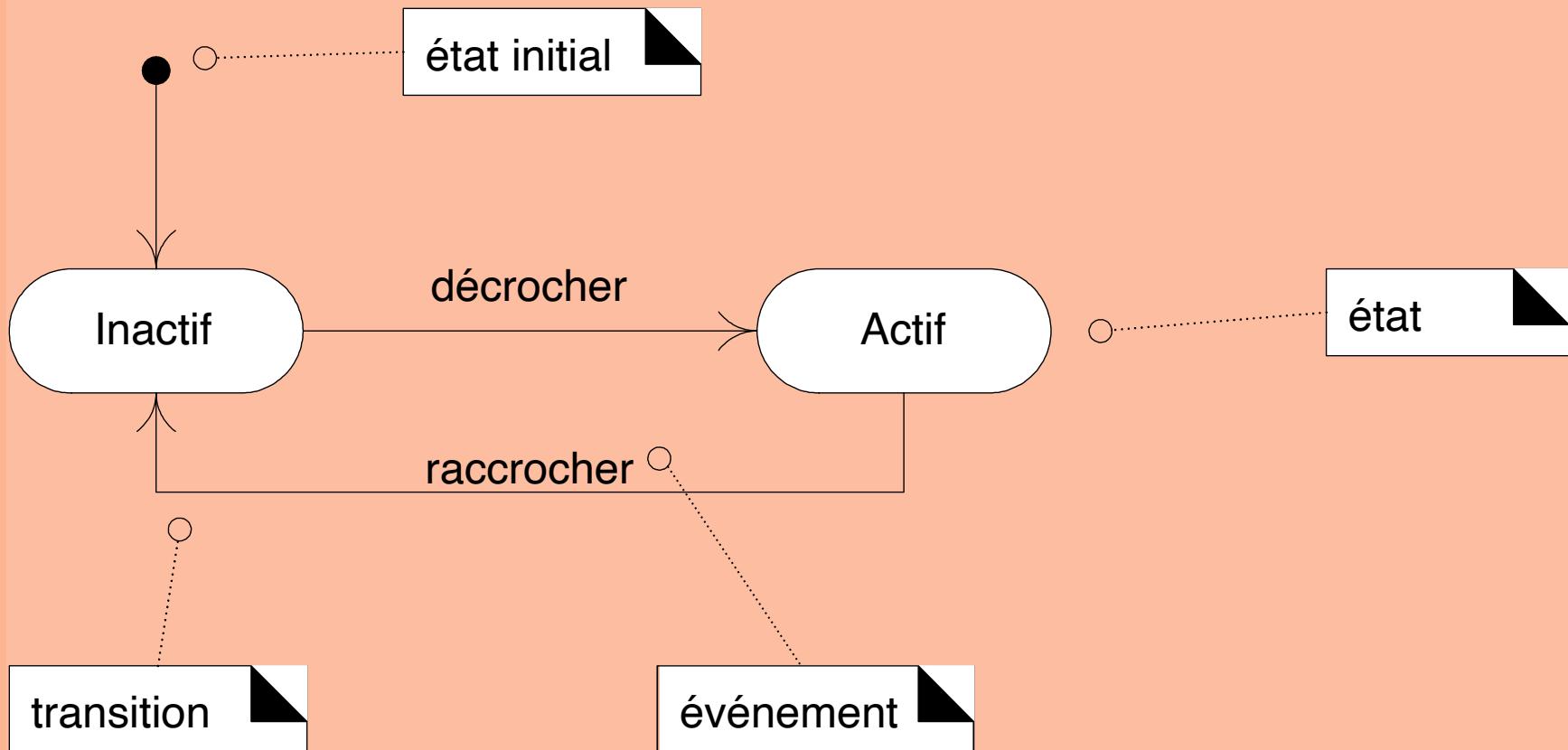
Notion préalable: Automate fini

# EXEMPLE D'AUTOMATE FINI

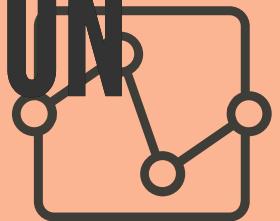


Téléphone

Created by Rudez Studio  
from the Noun Project



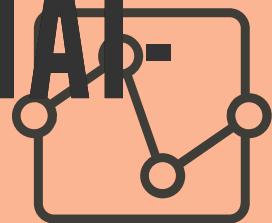
# ÉVÉNEMENT, ÉTAT ET TRANSITION



- Événement
  - occurrence d'un fait significatif ou remarquable
- État
  - la condition d'un objet à un moment donné, jusqu'à l'arrivée d'un nouvel événement
- Transition
  - relation état-événement-état
  - indique que l'objet change d'état

Created by Rudez Studio  
from the Noun Project

# OBJETS ÉTAT-DÉPENDANTS ET ÉTAT-INDÉPENDANTS



Created by Rudez Studio  
from the Noun Project

- Un objet répondant de la même manière à un événement donné
  - état-indépendant (par rapport à l'événement )
- Un objet répondant différemment, selon son état, à un événement donné
  - état-dépendant

# MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS



- Équipement physiques contrôlés par des logiciels

Created by Rudez Studio  
from the Noun Project



# MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS



Transactions et objets métier apparentés

Created by Rudez Studio  
from the Noun Project

- Vente, Commande, Paiement, etc.
- Exemple: annuler l'envoi d'un colis après son ramassage?  
Cela dépend de son « état » probablement...



# MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS



Objets qui changent de rôle, comme une personne...

Created by Rudez Studio  
from the Noun Project

- état de civil ou de militaire
- résident temporaire, résident permanent, citoyen



# MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS

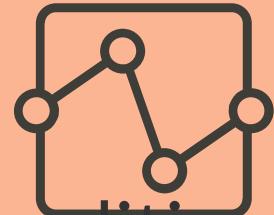


Created by Noun Project  
from the Noun Project

- Protocoles de communication
  - p.ex. en TCP, si le gestionnaire du protocole est dans l'état « fermé », un message « close » sera ignoré
- Flot page/fenêtre IHM
  - séquence légale des pages Web ou des fenêtres, souvent complexe
- Contrôleurs de flot ou de sessions
  - objets « session » p.ex. application Web
- Ordre des opérations système d'un cas d'utilisation
  - créerNouvelleVente , saisirArticle , terminerVente , etc.
- Gestion d'un événement individuel dans une fenêtre d'une IHM
  - séquences légales d'une fenêtre ou d'un formulaire (coller n'est pas valide si presse-papiers est nul)

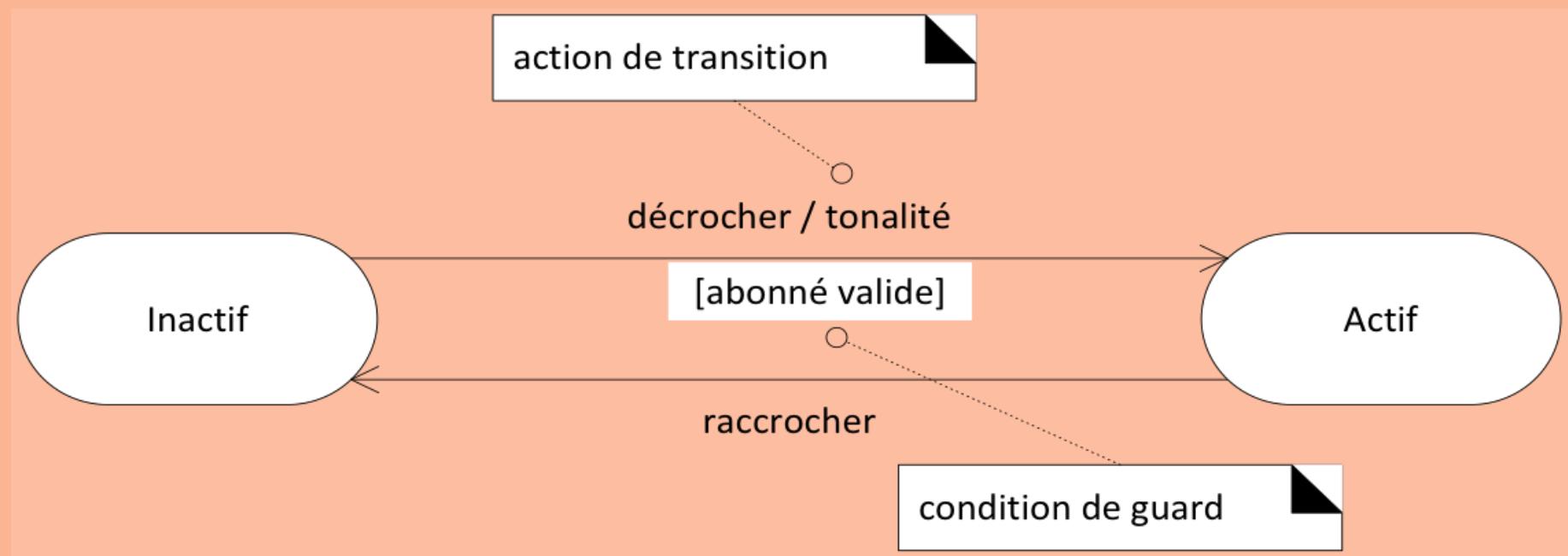


# NOTATION ADDITIONNELLE

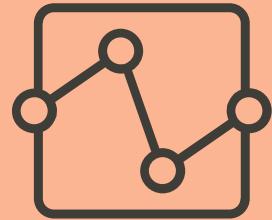


Transitions peuvent avoir des actions et des conditions de guard

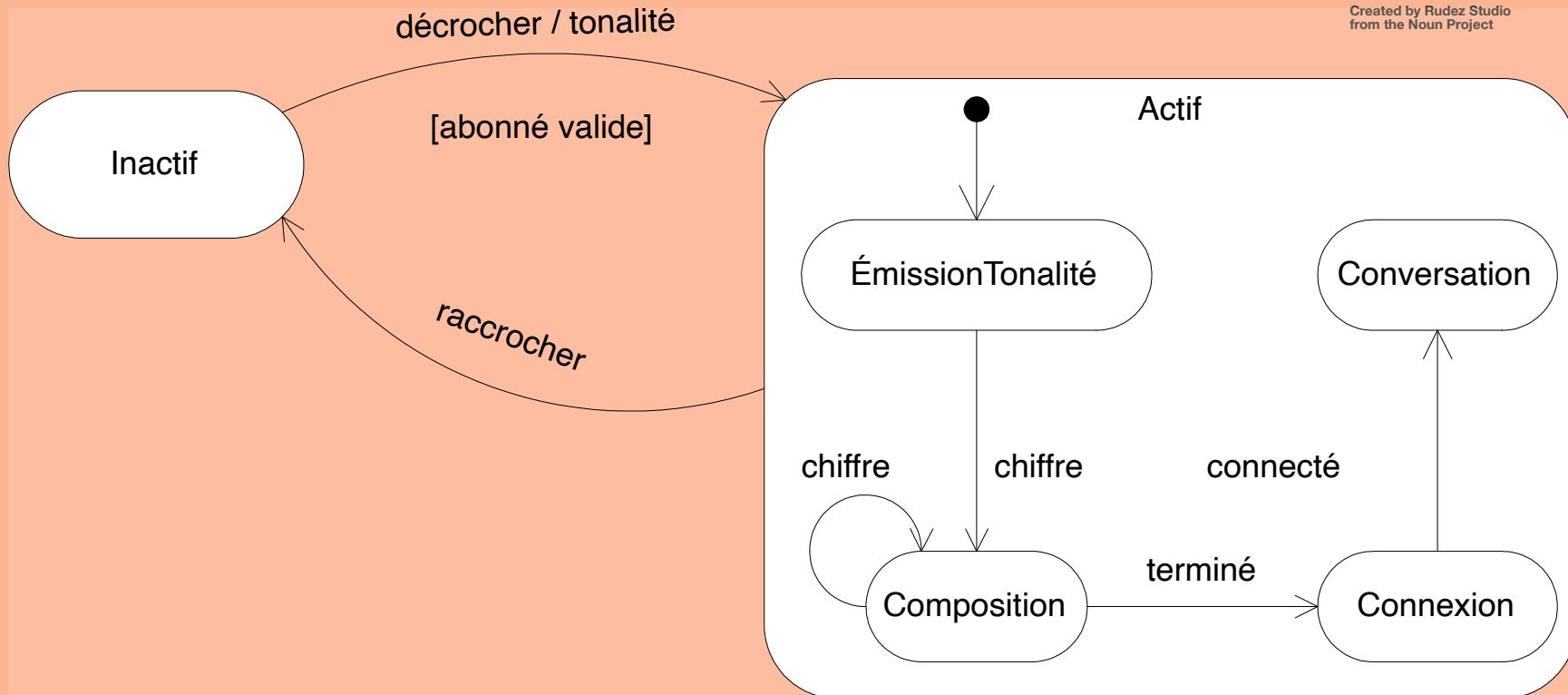
Created by Rudez Studio  
from the Noun Project



# NOTATION ADDITIONNELLE

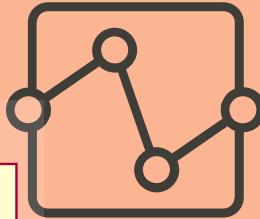
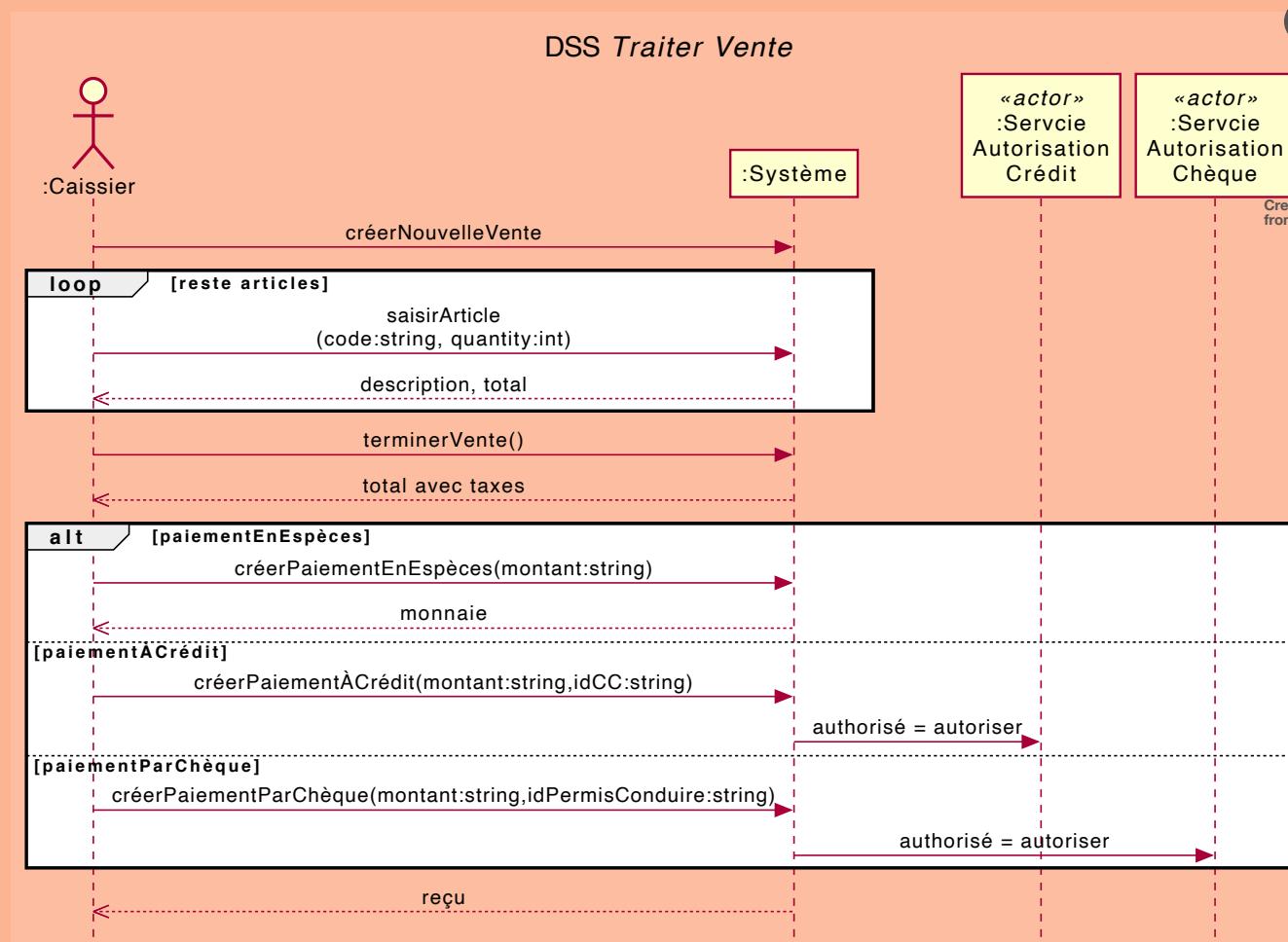


## États imbriqués



- Pouvez-vous réaliser le diagramme de classe correspondant?

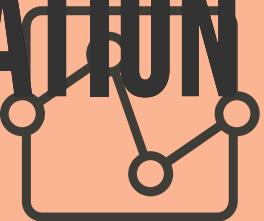
# \*AUTOMATE FINI ET CAS D'UTILISATION 1/2



Created by Rudez Studio  
from the Noun Project

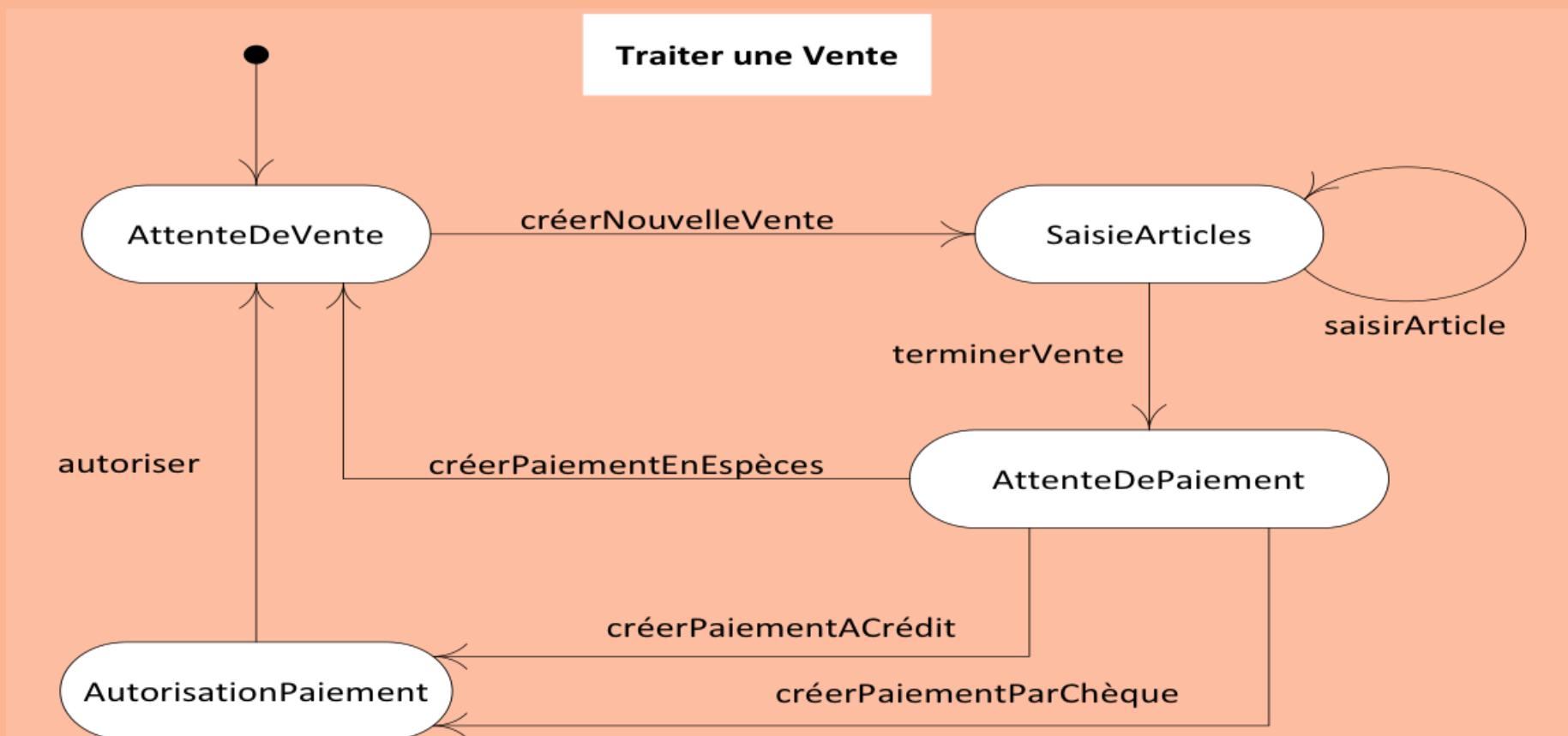
- Diagramme d'état correspondant?

# \*AUTOMATE FINI ET CAS D'UTILISATION

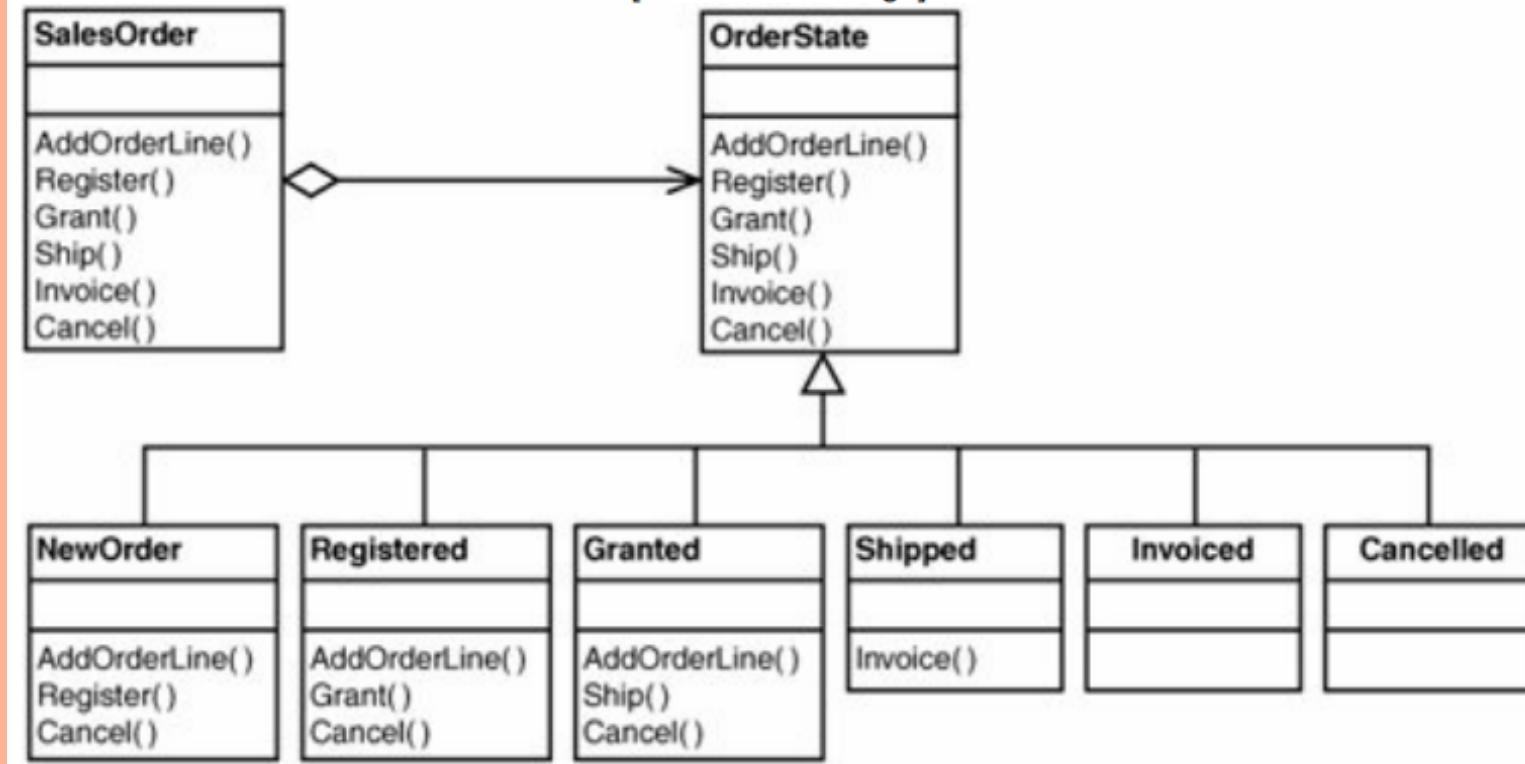
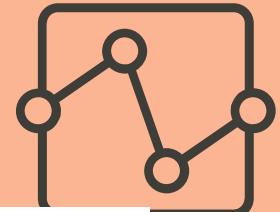


## 2/2

Created by Rudez Studio  
from the Noun Project



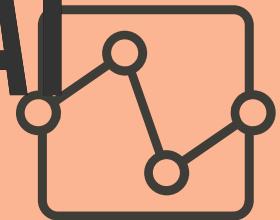
# ÉTAT D'UNE COMMANDE



Réf: Applying Domain-Driven Design and Patterns:  
With Examples in C# and .NET



# EXERCICES DIAGRAMME D'ÉTAT

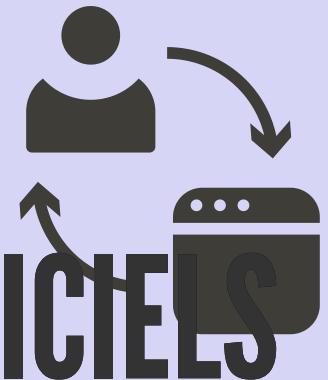


Created by Rudez Studio  
from the Noun Project

- Téléphone intelligent
- Vidéo projecteur
- Guichet automatique
- CU29-Annuler un service
- CU30-Confirmer une visite supervisée
- CU31-Confirmer des échanges de garde
- CU32-Rédiger une note d'observation
- CU33-Corriger une note d'observation

# LOG210 SÉANCE #10

## ANALYSE ET CONCEPTION DE LOGICIELS



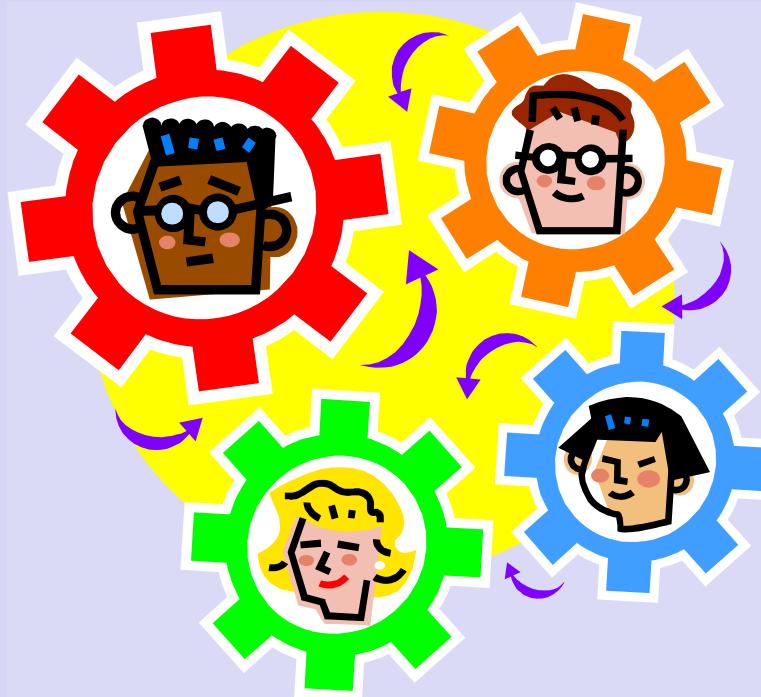
1. Affinement du MDD
2. Diagramme d'interaction
3. Diagramme d'état
4. Diagramme d'activité



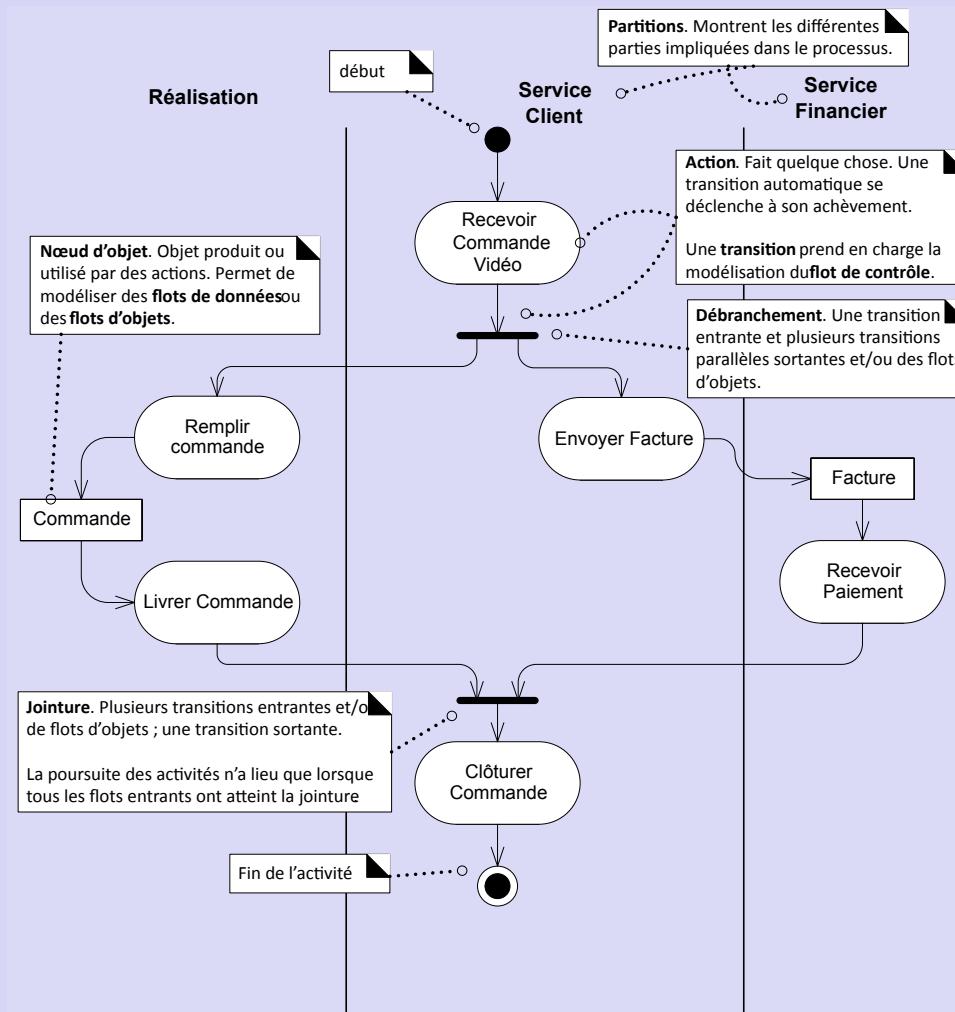
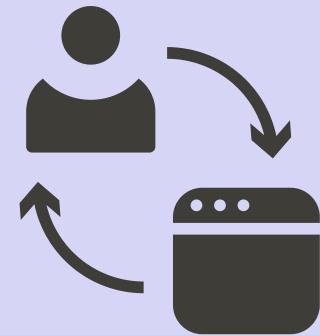
16.01m

# DIAGRAMMES D'ACTIVITÉS

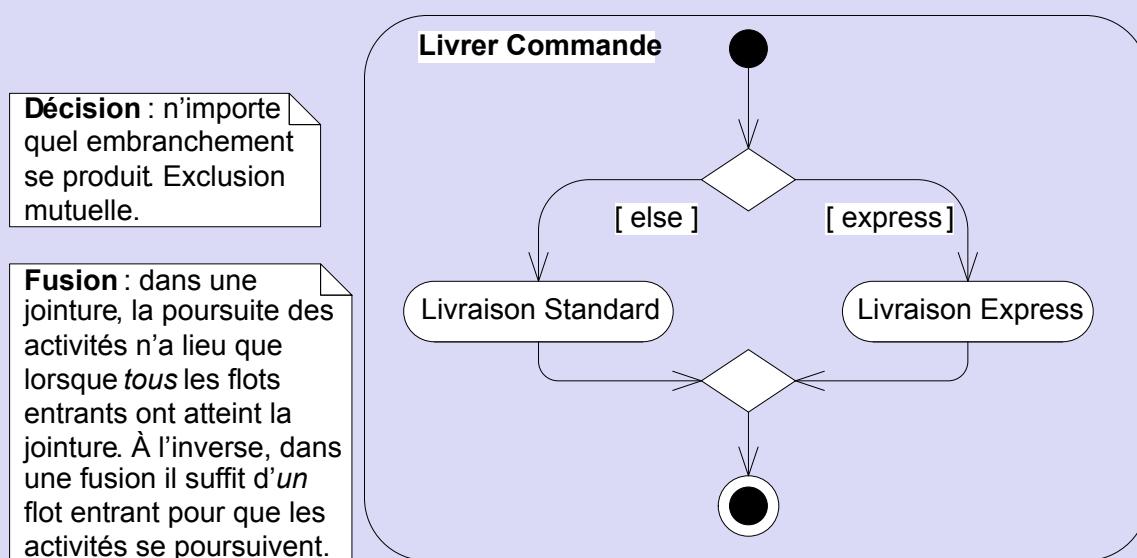
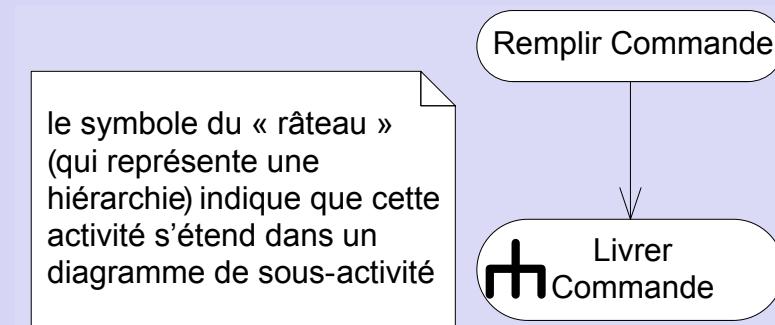
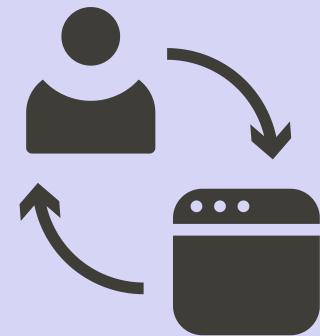
- Exposer les activités séquentielles et parallèles d'un processus.



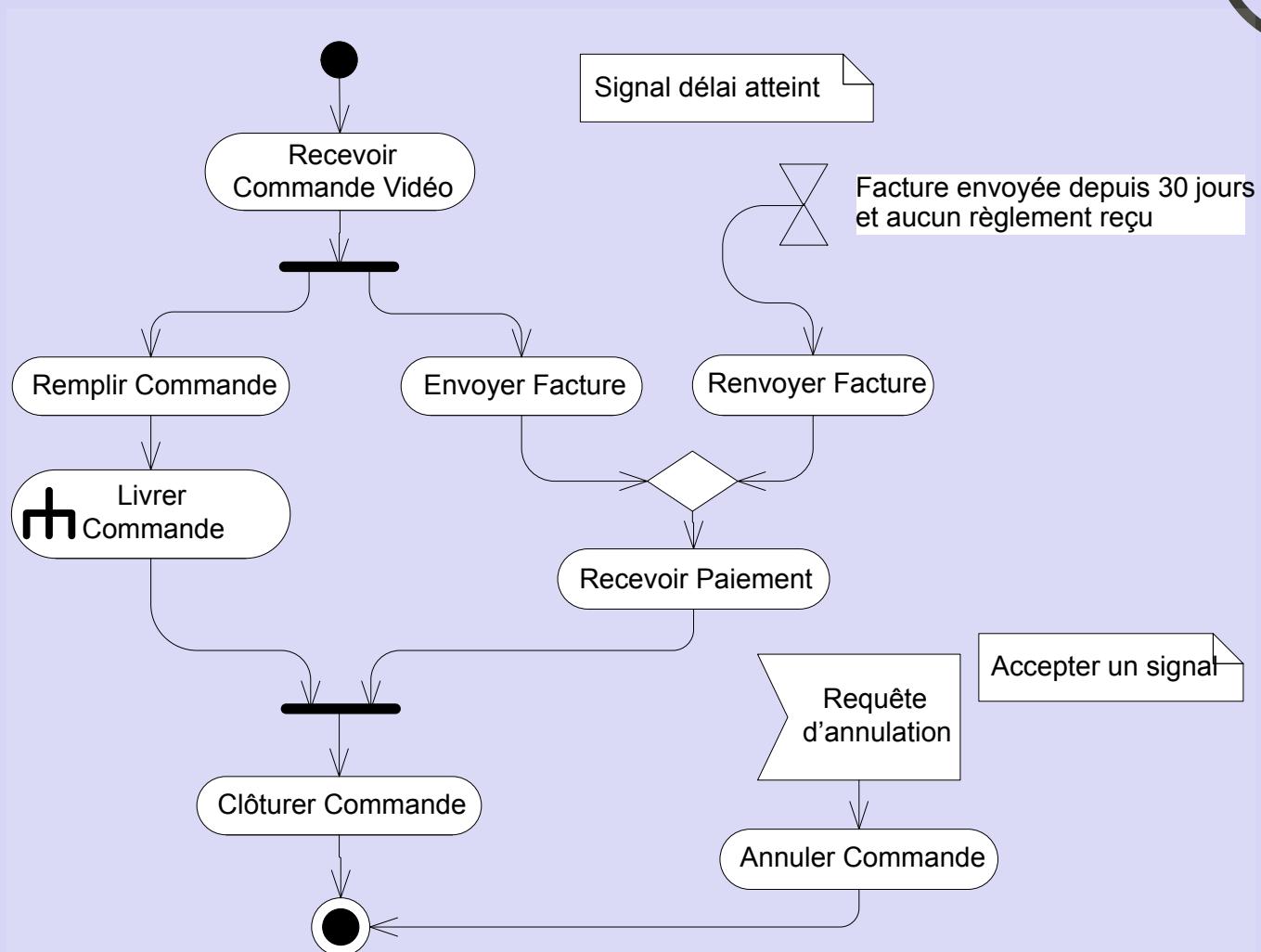
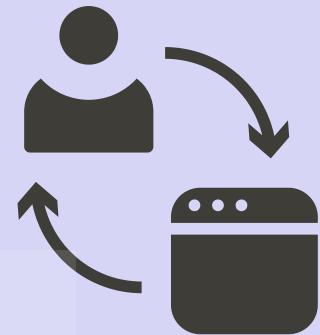
# DIAGRAMMES D'ACTIVITÉS



# COMPLÉMENTS SUR LA NOTATION



# SIGNALS

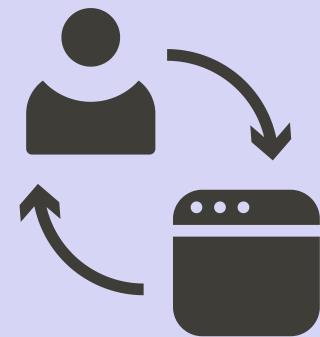
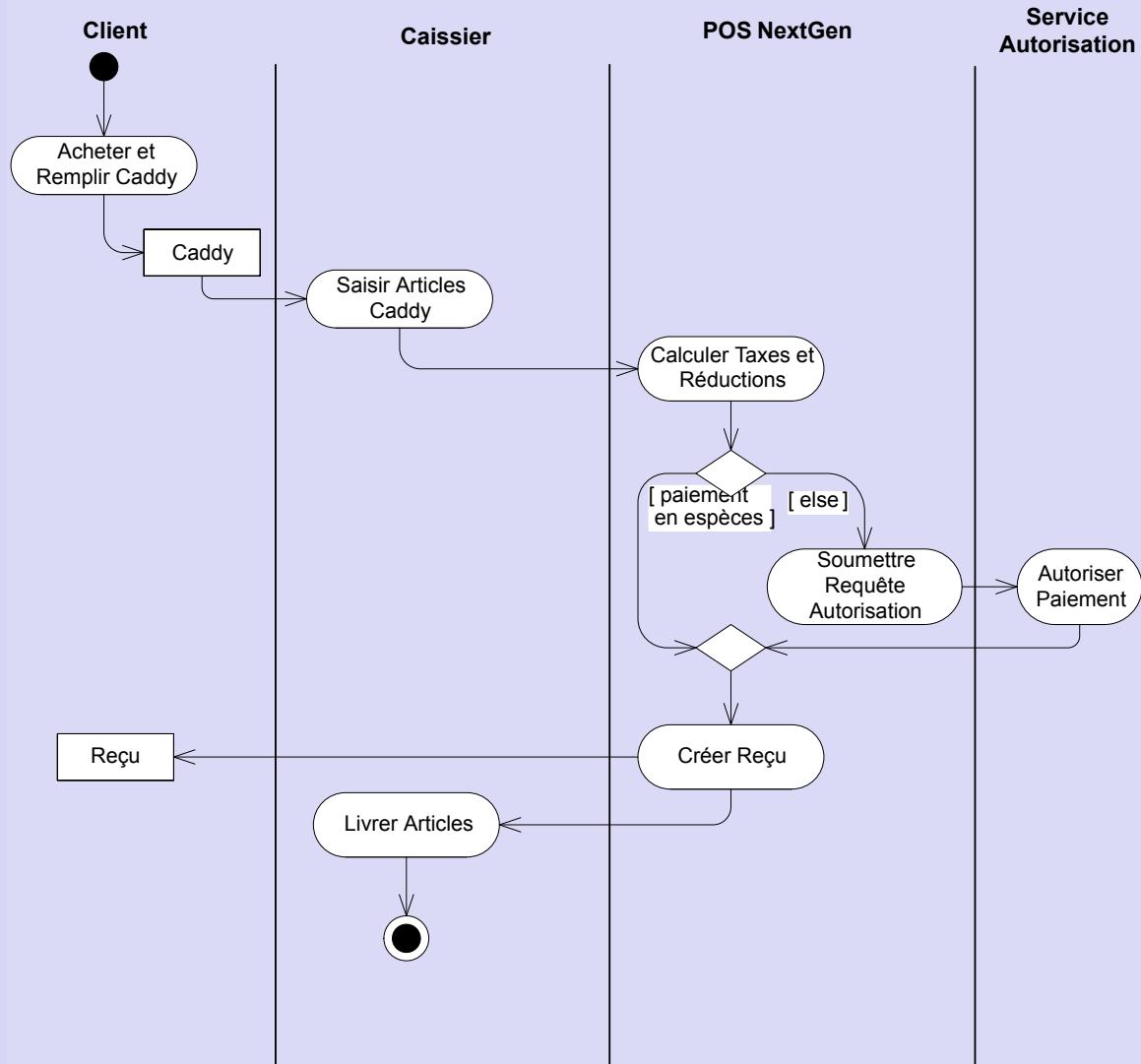


# LIGNES DIRECTRICES – MODÉLISATION D'ACTIVITÉS

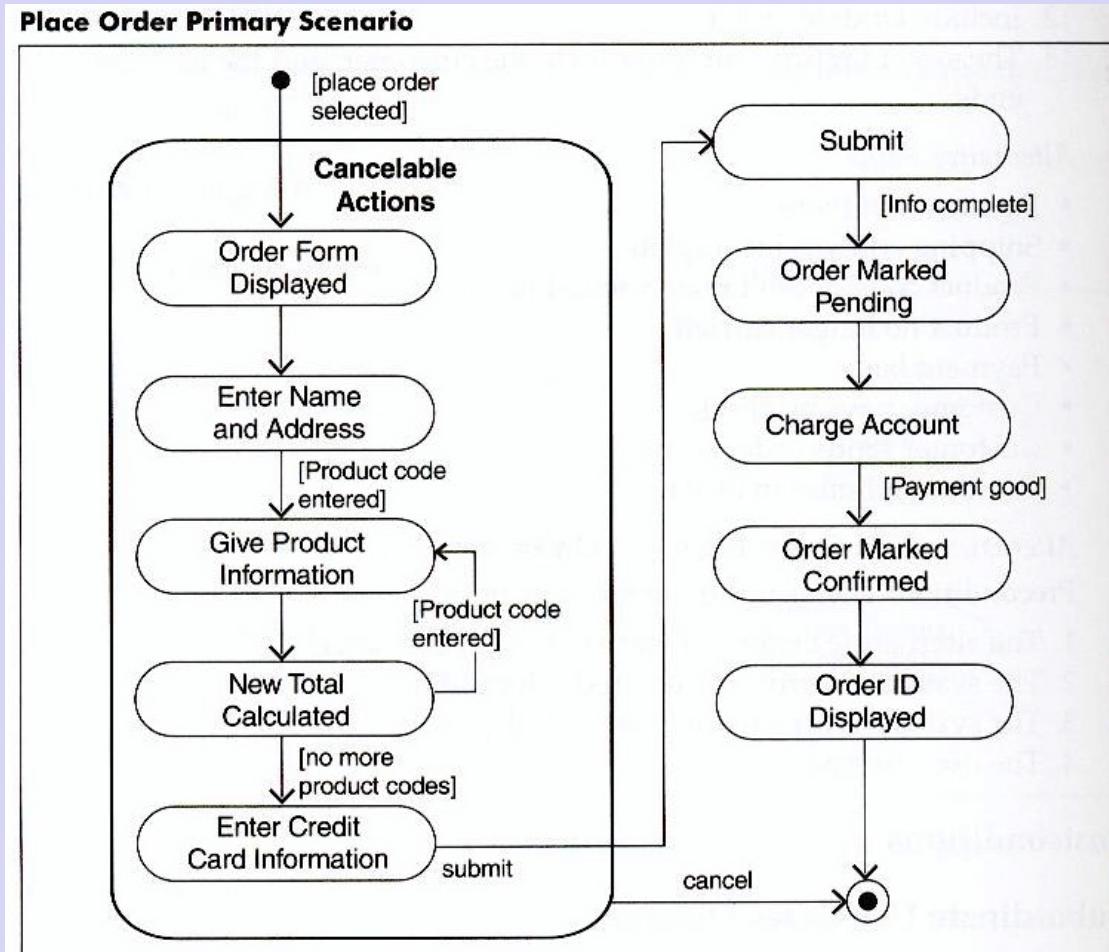
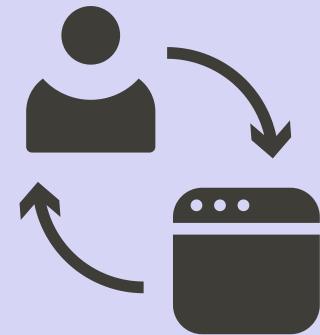
- Utile pour les processus très complexes
  - impliquant plusieurs parties
  - impliquant des activités séquentielles
- ■ impliquant des processus en parallèles
- Utilisez les niveaux d'abstraction («rateau» et sous-activité) afin de gérer la complexité
  - Niveaux 0, 1, 2, etc.
  - Dans un diagramme, veillez à ce que les nœuds d'action soient à peu près équivalents en ce qui concerne leur propre niveau d'abstraction.
  - p.ex. au niveau 0, « Livrer Commande » et « Calculer TVA » ne sont pas cohérents - « Livrer Commande » et « Envoyer Facture » le sont



# MODÉLISATION DE TRAITER UNE VENTE

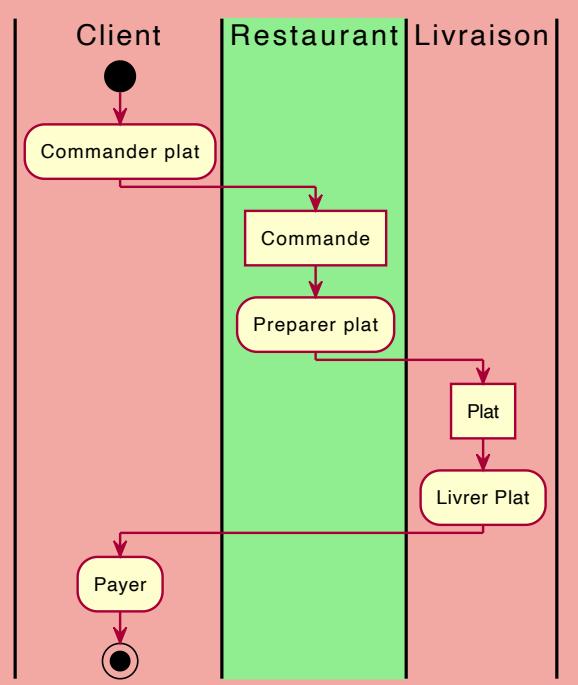
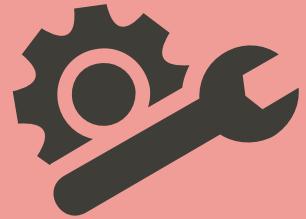


# D'AUTRES EXEMPLES



- Applying Use Cases, 2nd ed., Schneider & Winters, Addison Wesley, 2001.

# OUTILS PLANTUML



```

|Client|
start
:Commander plat;
|#lightgreen|Restaurant|
:Commande]
:Preparer plat;
|Livraison|
:Plat]
:Livrer Plat;
|Client|
:Payer;
stop

```

Exemple : Dynamique GitHub Classroom

# EXERCICE: RETOUR DE VOITURE LOUÉ



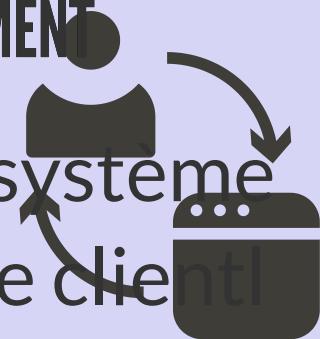
Esquissez le diagramme d'activités lors de la réception de voitures louées (après la location) dans une compagnie. Pour le diagramme, faites attention à la **notation UML**: cela comprend les objets (pour la voiture et pour la facture), le début et la fin de l'activité, les débranchements, les jointures, les décisions et les fusions.

## Scénario

- Le client rend la voiture et les clés.
- Le réceptionniste note le kilométrage et le niveau d'essence pour calculer la facture.
- Le client paye sa location, selon le montant sur la facture et part après.
- L'agent inspecte la voiture pour la propreté. Si elle n'est pas assez propre, alors l'agent doit laver, rincer et sécher l'extérieur et nettoyer l'intérieur. Ce travail devrait commencer le plus vite possible, après que le réceptionniste ait fini de noter les informations pour la facture.
- Les rôles sont le \_\_\_\_\_ le \_\_\_\_\_ (qui gère la documentation et le paiement de la location) et \_\_\_\_\_ (qui gère le traitement des voitures avant la prochaine location).



# \*EXERCICES - MODÉLISER D'UNE DYNAMIQUE DE STATIONNEMENT



- Client arrive, entre sa carte de crédit, le système enregistre la carte et la remet au client, le client reprend sa carte, le système ouvre la barrière, le client entre pour se stationner, le système ferme la barrière. Le client se présente à la sortie, présente sa carte de crédit, le système imprime le recu, le client reprend sa carte de crédit, prend le recu, le système ouvre la barrière, le client peut alors sortir, le système détecte la sortie et ferme la barrière.

# EXERCICES DIAGRAMME D'ACTIVITÉ

- Retour de voiture louée
- Recette de cuisine
- Retrait au guichet automatique
- Ordinateur de plongée
- Demander un remplacement
- Processus d'achat sur le web
- Vendre au comptoir



<https://github.com/yvanross/LOG210-exercices>

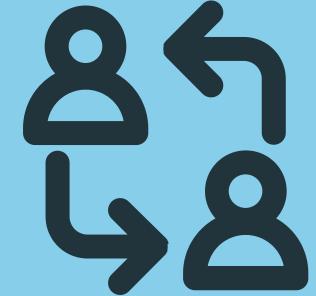
# RÉSUMÉ

1. Affinement du MDD
2. Diagramme d'interaction
3. Diagramme d'état
4. Diagramme d'activité



# SÉANCE #10

## RÉTROACTION: PAGE D'UNE MINUTE



Created by Prithvi  
from the Noun Project

1. Quels sont les deux [trois, quatre, cinq] plus importants [utiles, significatives, surprenantes, dérangeantes] choses que vous avez apprises au cours de cette session?
2. Quelle (s) question (s) reste (s) en tête dans votre esprit?
3. Y a-t-il quelque chose que tu n'as pas compris?

<https://1drv.ms/u/s!An6-F73ulxAOhVyiCB46jTeINVLS>

