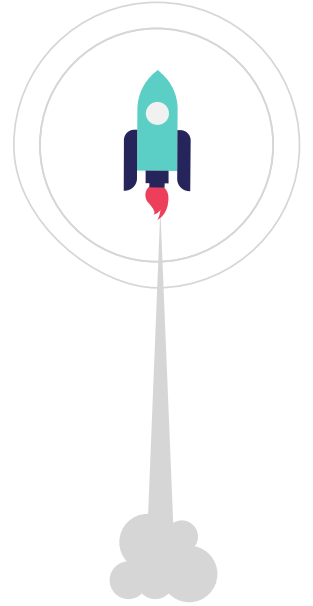




# UML




Par Mariam BENLLARCH

# La Notion UML



# Introduction au langage de modélisation UML

**UML:**

<b>U</b>		<b>Unified</b>	
<b>M</b>		<b>Modeling</b>	<b>=</b>
<b>L</b>		<b>Language</b>	

Langage de  
modélisation  
unifié

# Introduction au langage de modélisation UML

**UML:**

**U** → **Unified**

**M** → **Modeling**

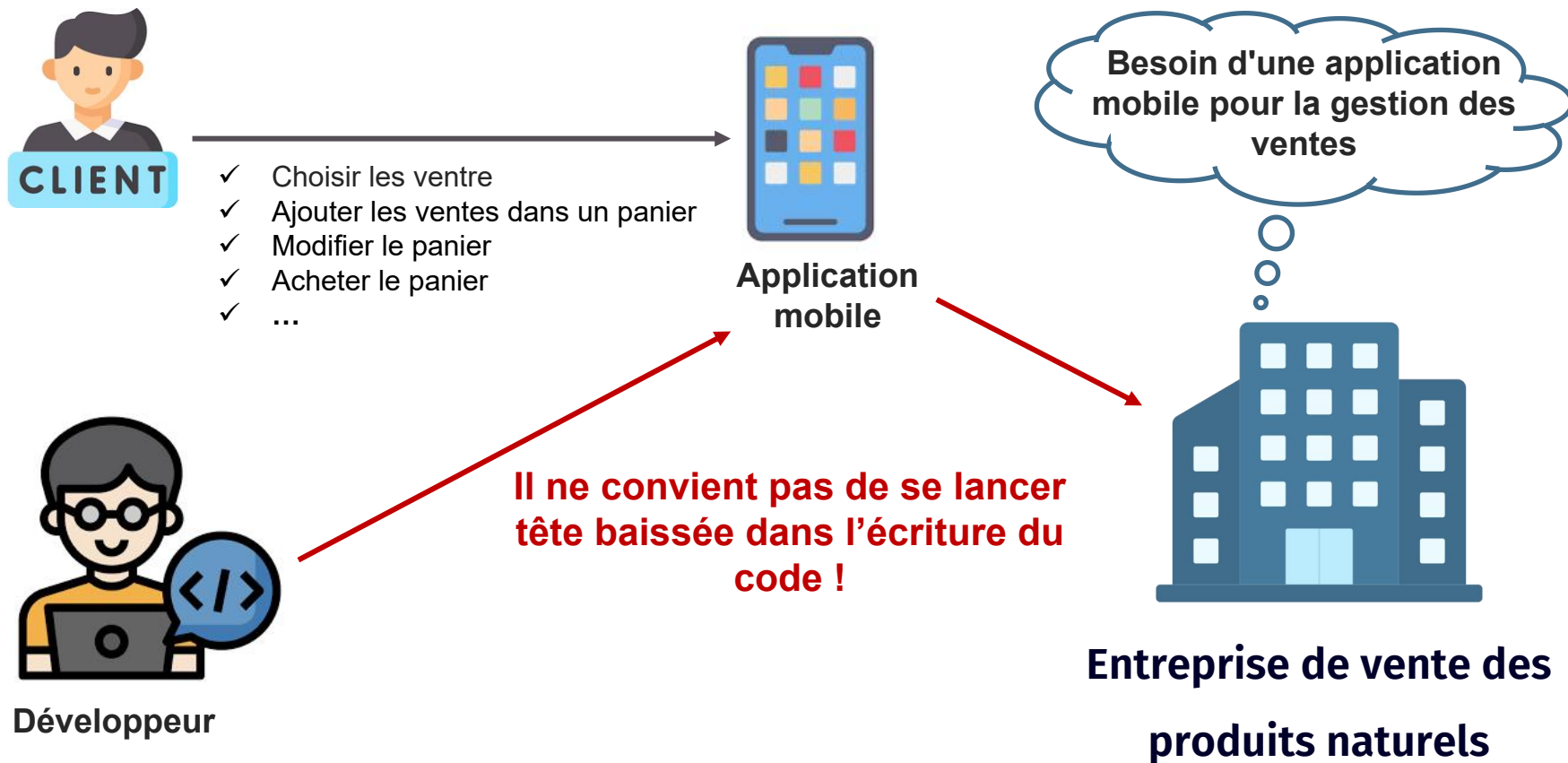
**L** → **Language**

Langage de

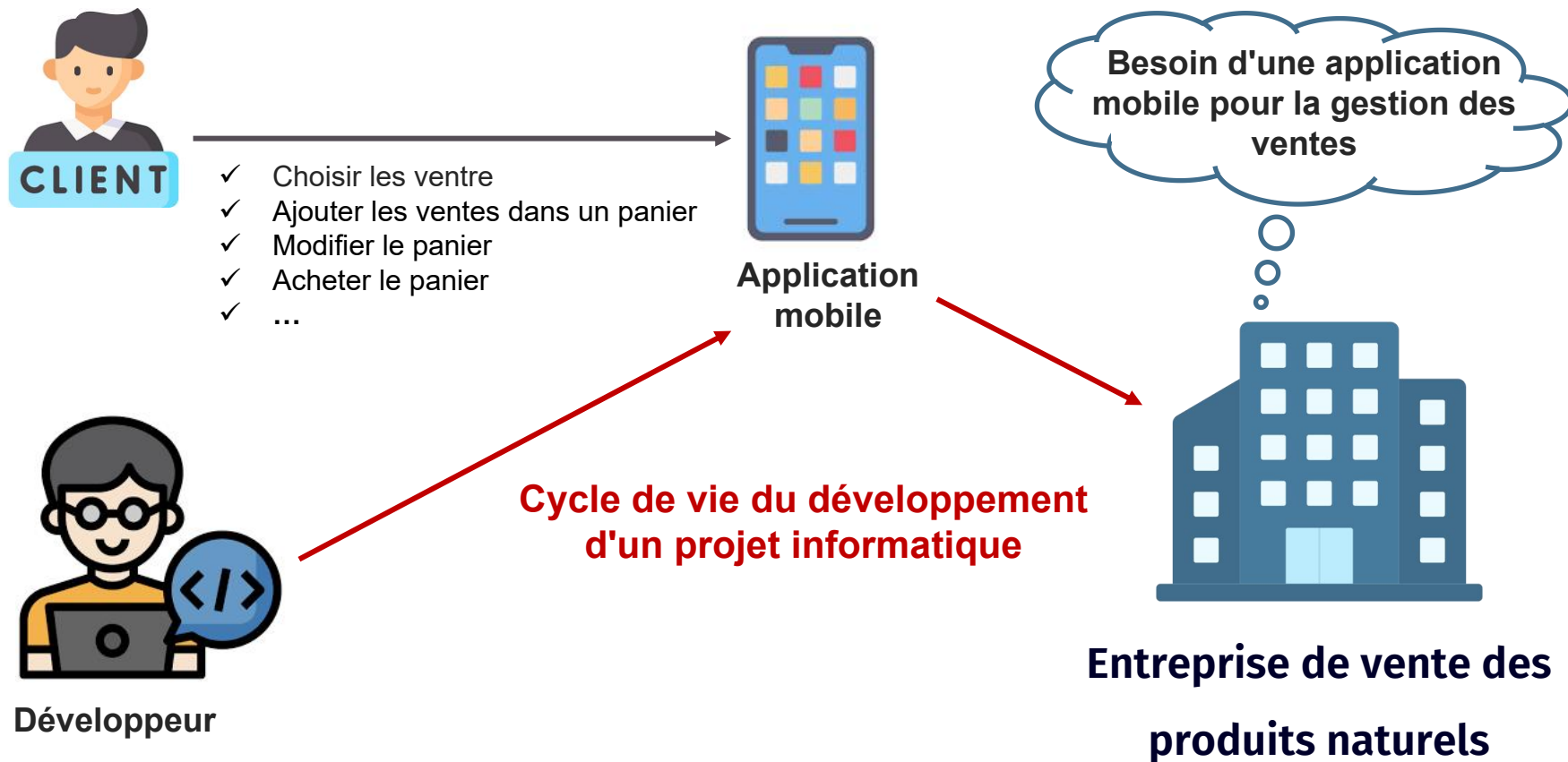
= **modélisation**

unifié

# C'est quoi la modélisation ?



# C'est quoi la modélisation ?



# C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

# C'est quoi la modélisation ?

Développeur



Application mobile



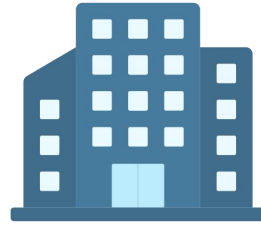
Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** **Etude de l'existant**, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.



# C'est quoi la modélisation ?

**Existant dans le système de l'entreprise**



- ✓ Les produits et les types de produits
- ✓ Les matériels informatiques, Les logiciels, et les réseaux
- ✓ La gestion des ventes
- ✓ ...

# C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, **expression des besoins**, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

# C'est quoi la modélisation ?

## Besoin des clients



- ✓ Choisir les ventres
- ✓ Ajouter les ventes dans un panier
- ✓ Modifier le panier
- ✓ Acheter le panier
- ✓ ...

## Besoin de l'entreprise



- ✓ La méthode de paiement
- ✓ L'organisation des produits dans l'application
- ✓ ...

# C'est quoi la modélisation ?

Développeur



Application mobile

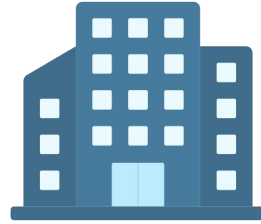


Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, **définition des limites.**
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

# C'est quoi la modélisation ?

## Les limites de l'entreprise



- ✓ Limites de ressource humaines
- ✓ Limites du budget
- ✓ Limites technologiques
- ✓ ...

# C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

# C'est quoi la modélisation ?

Développeur



Application mobile

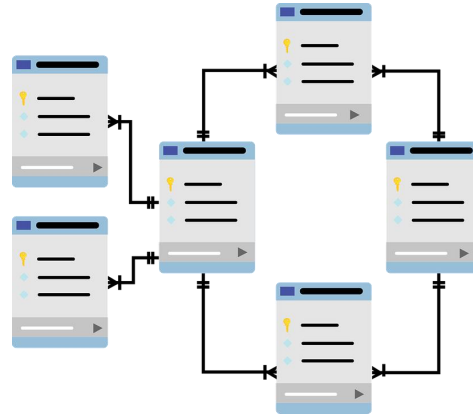


Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme **des modèles**.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

# C'est quoi la modélisation ?

## Modèle



Un modèle est une simplification de la réalité qui permet de mieux comprendre le système à développer (ex: un plan, une carte, un schéma électronique ...).



# C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :




- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme **des modèles**.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

# Pourquoi modéliser ?

- ✓ Pour visualiser le système comme il est ou comme il devrait l'être.
- ✓ Pour valider le modèle vis à vis des clients.
- ✓ Pour spécifier les structures de données et le comportement du système.
- ✓ Pour fournir un guide pour la construction du système.
- ✓ Pour documenter le système et les décisions prises.

# Introduction au langage de modélisation UML




**UML:**

<b>U</b>		<b>Unified</b>	
<b>M</b>		<b>Modeling</b>	<b>=</b>
<b>L</b>		<b>Language</b>	

**Langage de  
modélisation  
unifié**

# Introduction au langage de modélisation UML

**UML:**

<b>U</b>		<b>Unified</b>	
<b>M</b>		<b>Modeling</b>	<b>=</b>
<b>L</b>		<b>Language</b>	

**Langage de modélisation unifié**

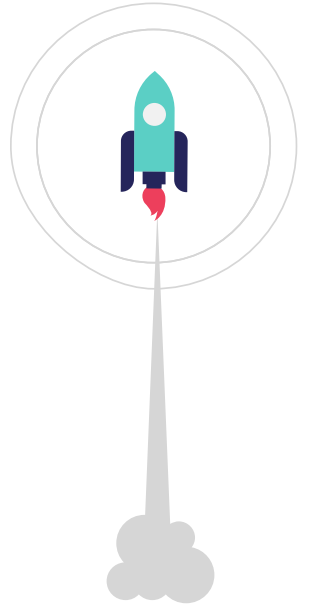
# UML est un langage

- ✓ UML n'est pas une méthode,
- ✓ UML est un langage de modélisation objet,
- ✓ UML est dans le domaine public, c'est une norme.

# UML un langage pour

- ✓ visualiser: chaque symbole graphique a une sémantique,
- ✓ spécifier: de manière précise et complète,
- ✓ construire: les classes, les relations SQL peuvent être générées automatiquement,
- ✓ documenter: les différents diagrammes, notes, contraintes, exigences seront présentés dans un document.

# Comment modéliser avec UML ?



# L' utilisation des diagrammes

- ✓ UML permet de définir et de visualiser un modèle, à l'aide de **diagrammes**.
- ✓ **Un diagramme** UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle.
- ✓ Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis).
- ✓ Un type de diagramme UML offre toujours la même vue d'un système (il véhicule une sémantique précise).
- ✓ Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.



# Les différents types de diagrammes UML

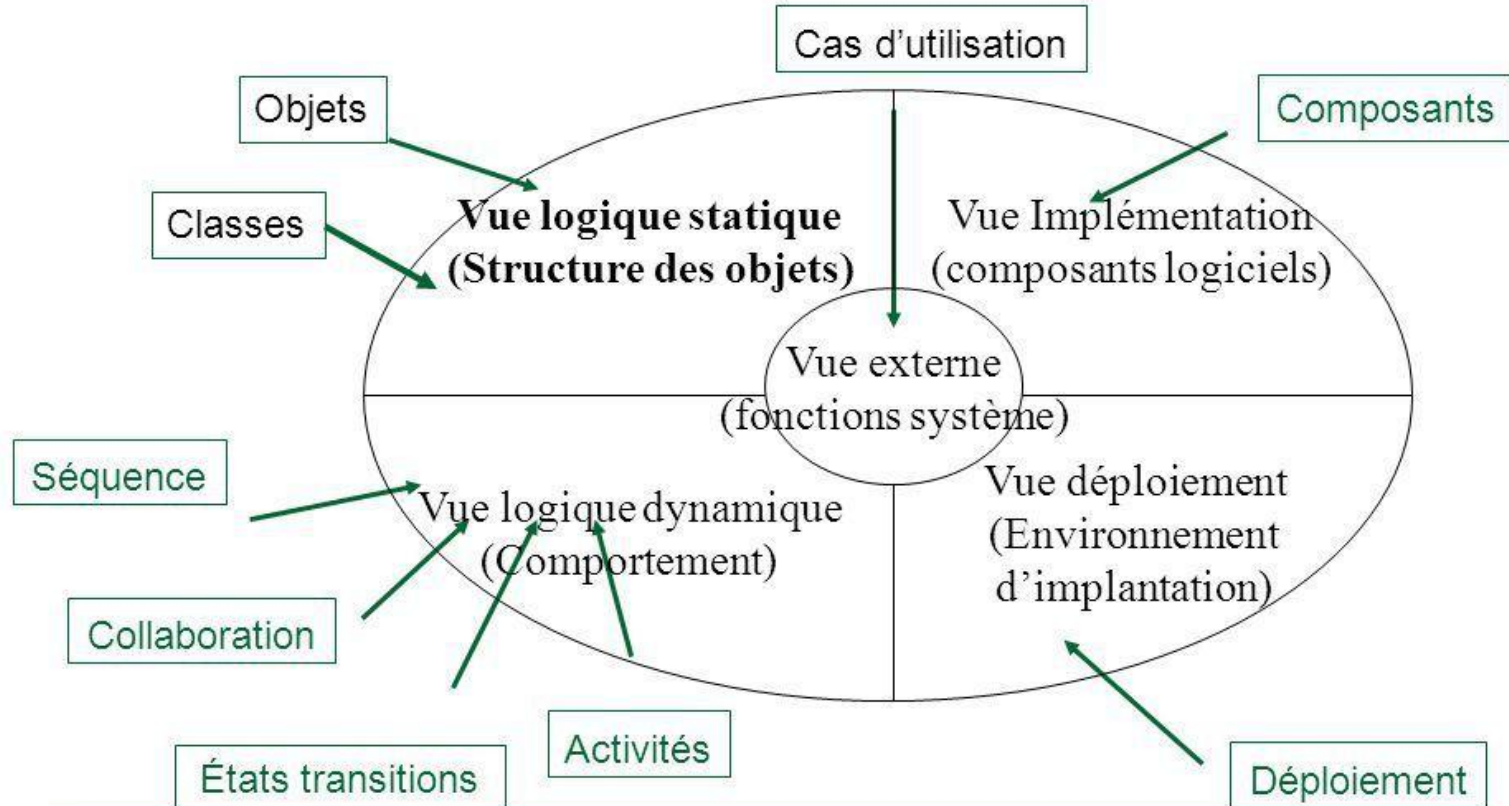
## ✓ **Vision statique du système :**

- ✓ diagramme de cas d'utilisation
- ✓ diagramme d'objets
- ✓ diagramme de classes
- ✓ diagramme de composants
- ✓ diagramme de déploiement

## ✓ **Vision dynamique du système :**

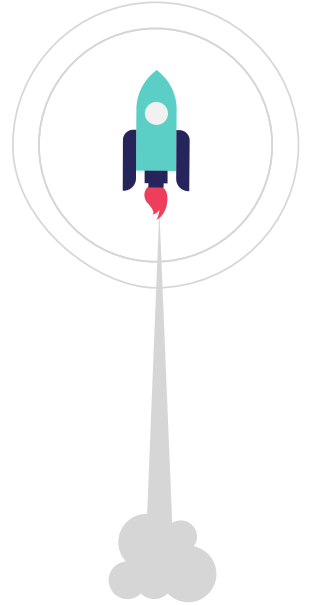
- ✓ diagramme de collaboration
- ✓ diagramme de séquence
- ✓ diagramme d'états-transitions
- ✓ diagramme d'activités

# Diagrammes et vues d'architecture UML



**Vision statique du système :**

**Diagramme de cas  
d'utilisation**



# Diagramme de Cas d'Utilisation

## Un diagramme de cas d'utilisation:

- Décrit:

- Les acteurs



- Les cas d'utilisation



- Le système




# Diagramme de Cas d'Utilisation

Un diagramme de cas d'utilisation:

- Décrit:

- **Les acteurs** 

- Les cas d'utilisation 

- Le système 

# Acteur

- Un acteur est une entité extérieure au système modélisé, et qui interagit directement avec lui.
- Un acteur est présenté par un petit bonhomme:



**Nom**  
**Acteur**

# Acteur

- Un acteur est une entité extérieure au système modélisé, et qui interagit directement avec lui.
- Un acteur est présenté par un petit bonhomme.
- Les principaux acteurs sont les utilisateurs du système.
- En plus des utilisateurs, les acteurs peuvent être :
  - Des logiciels déjà disponibles à intégrer dans le projet ;
  - Des systèmes informatiques externes au système mais qui interagissent avec lui ;
  - tout élément extérieur au système et avec lequel il interagit.



**Nom**  
**Acteur**

# Acteur

## Rôles et personnes physiques

- Un acteur correspond à un rôle, pas à une personne physique :
- Une même personne physique peut être représentée par plusieurs acteurs si elle a plusieurs rôles.
- Si plusieurs personnes jouent le même rôle vis-à-vis du système, elles seront représentées par un seul acteur.



**Nom  
Acteur**



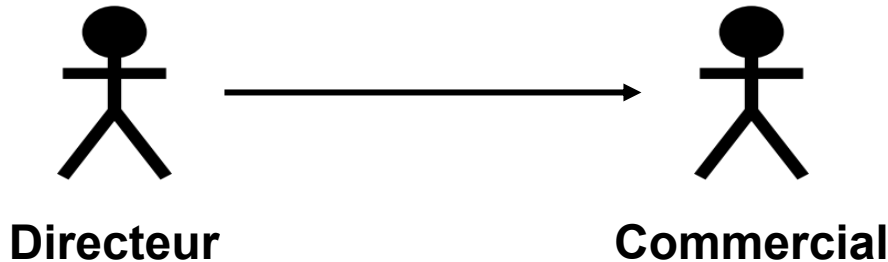
# Acteur

## Relations entre acteurs

- Il n'y a qu'un seul type de relation possible entre acteurs : la relation de généralisation.

### Exemple:

Un directeur est une sorte de commercial : il peut faire avec le système tout ce que peut faire un commercial, plus d'autres choses



# Diagramme de Cas d'Utilisation

## Un diagramme de cas d'utilisation:

- Décrit:

- Les acteurs



- **Les cas d'utilisation**



- Le système



# Cas d'utilisation

- Un cas d'utilisation est un moyen de représenter les différentes possibilités d'utiliser un système.
- Il exprime toujours une suite d'interactions entre un acteur et l'application.
- Il définit une fonctionnalité utilisable par un acteur.

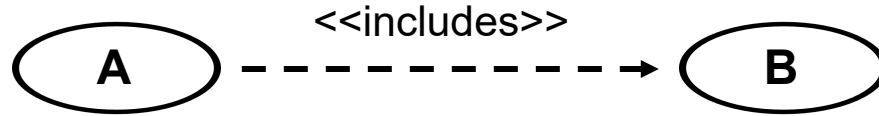


**Le cas  
d'utilisation**

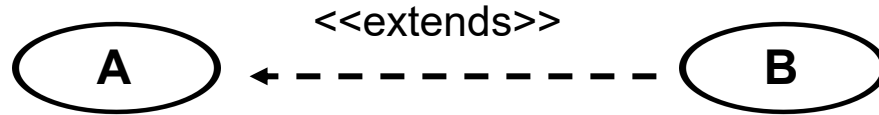
# Cas d'utilisation

## Relations entre les cas d'utilisations

1) **Inclusion** : B est une partie obligatoire de A et on lit A inclut B (dans le sens de la flèche).



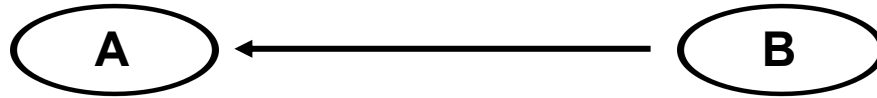
2) **Extension** : B est une partie optionnelle de A et on lit B étend A (dans le sens de la flèche).



# Cas d'utilisation

## Relations entre les cas d'utilisations

- 3) **Généralisation** : le cas A est une généralisation du cas du cas B et on lit B est une sorte de A.



# Diagramme de Cas d'Utilisation

## Un diagramme de cas d'utilisation:

- Décrit:

- Les acteurs



- Les cas d'utilisation



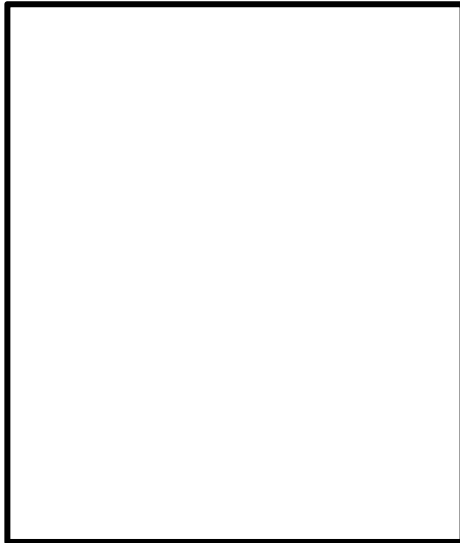
- **Le système**



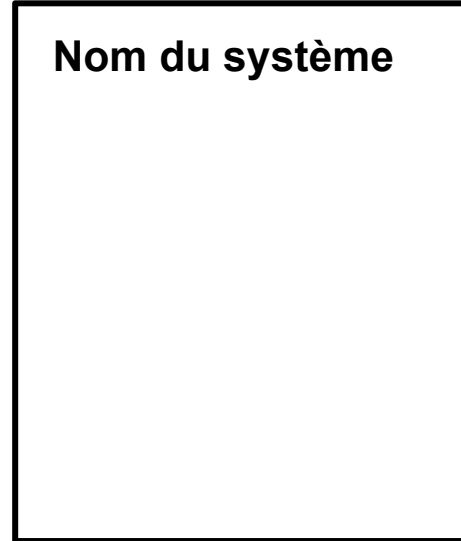
# Systeme

- Le système définit l'application informatique, il ne contient donc pas les acteurs, mais les cas d'utilisation et leur associations.

**Nom du système**



**Nom du système**



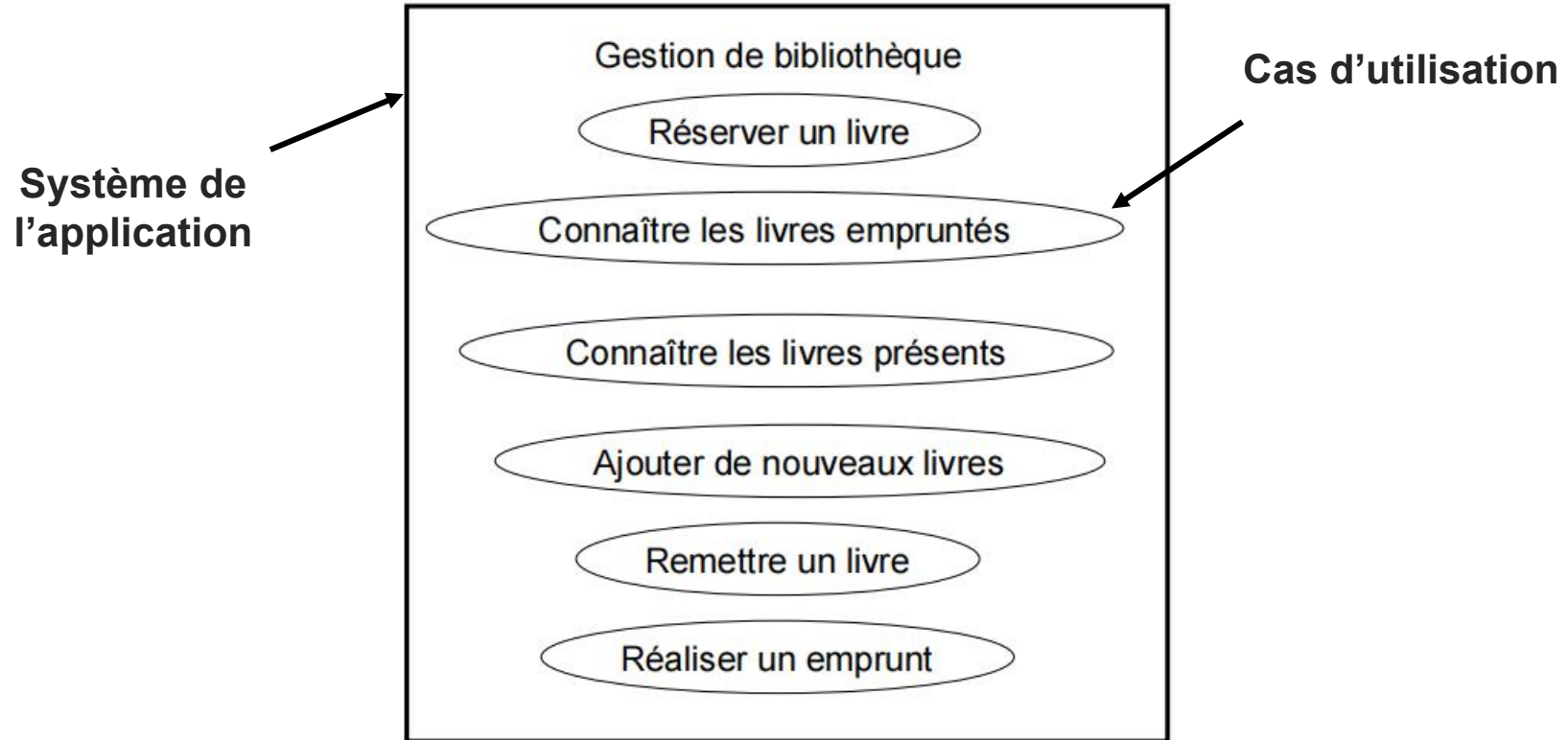
# Diagramme de Cas d'Utilisation

**Exemple:**

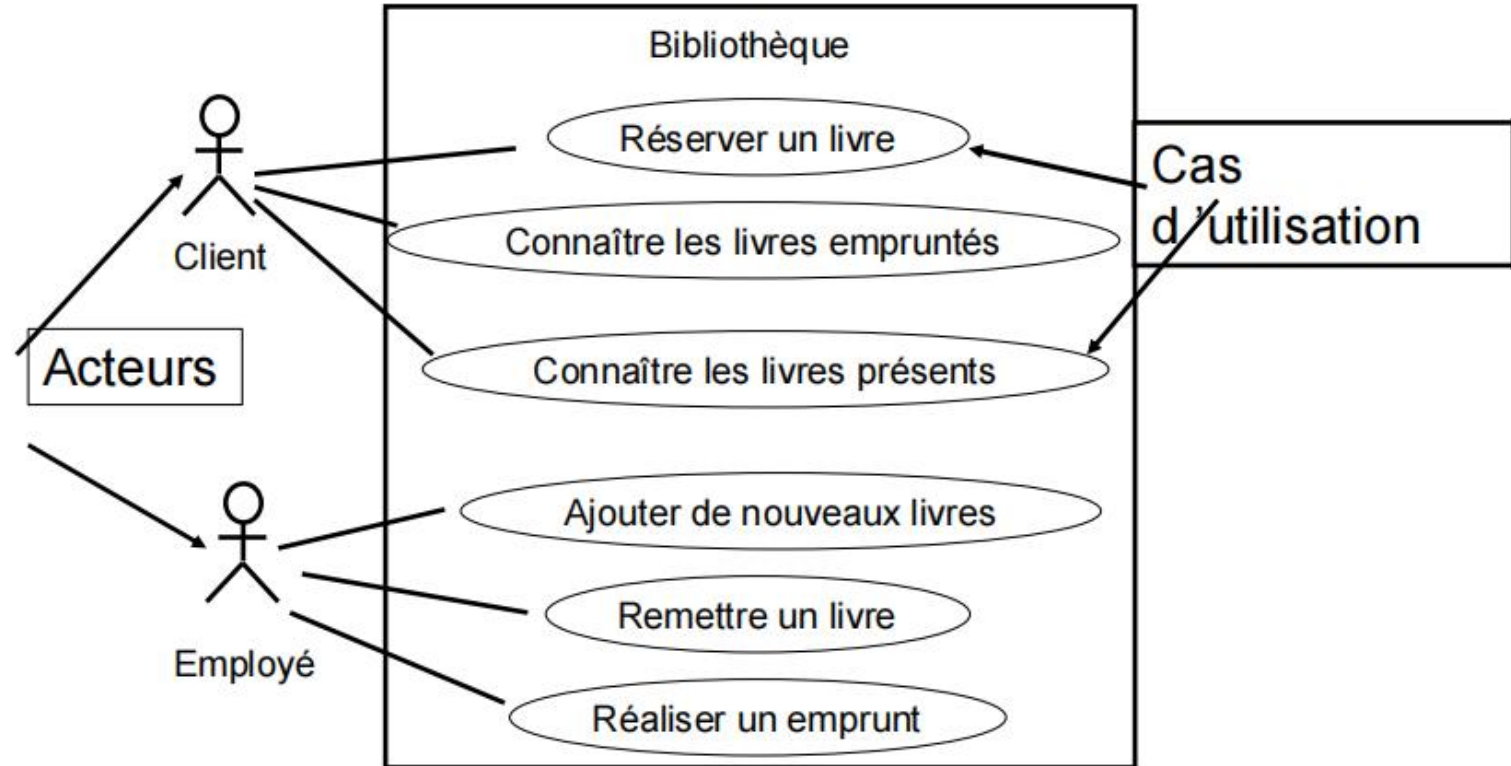
**Une application pour la gestion d'une  
bibliothèque**



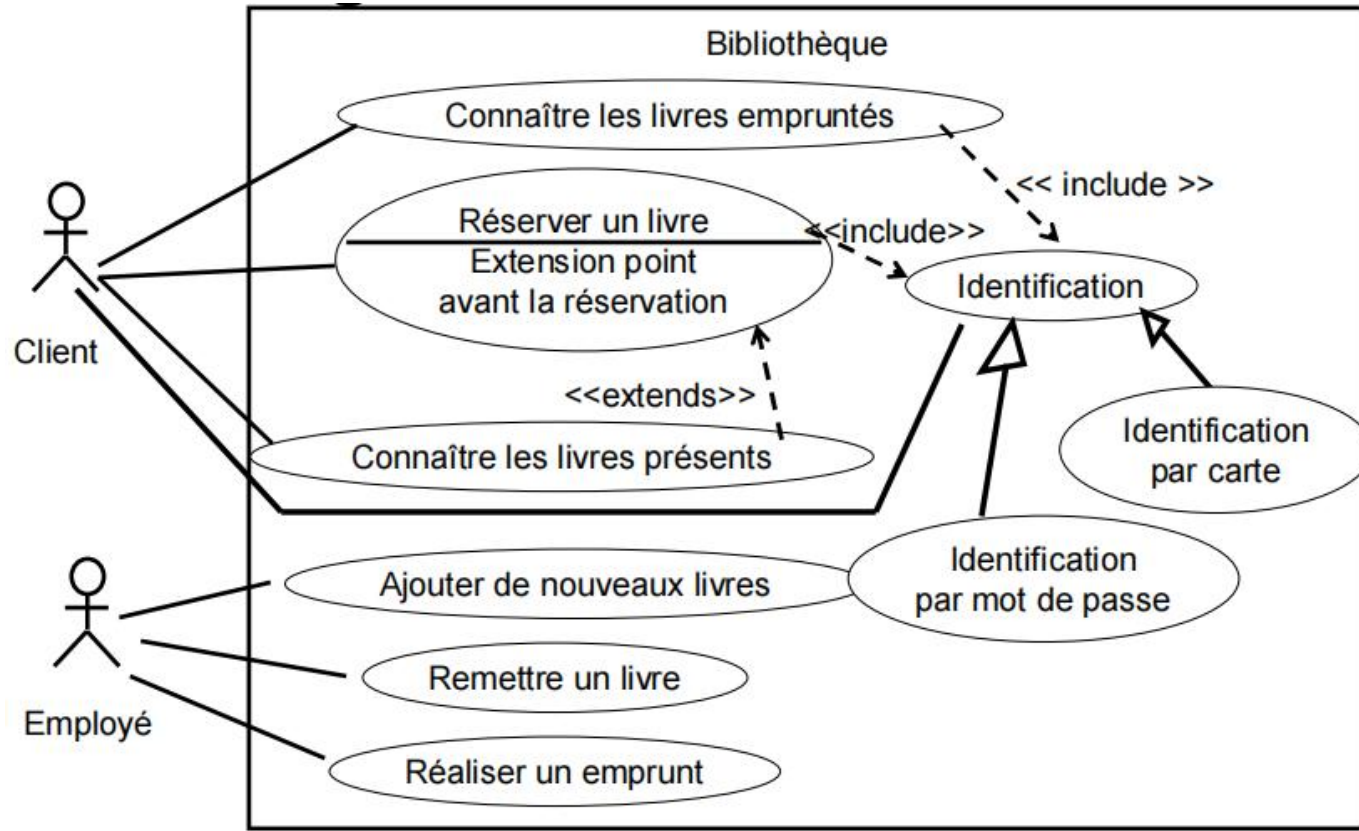
# Une application pour la gestion d'une bibliothèque



# Une application pour la gestion d'une bibliothèque



# Une application pour la gestion d'une bibliothèque



# Diagramme de Cas d'Utilisation

**Exercices**

# Exercice 1:

Dans un magasin, un commerçant dispose d'un système de gestion de son stock d'articles, dont les fonctionnalités sont les suivantes :

- Edition de la fiche d'un fournisseur,
- Possibilité d'ajouter un nouvel article (dans ce cas, la fiche fournisseur est automatiquement éditée. Si le fournisseur n'existe pas, on peut alors le créer),
- Edition de l'inventaire. Depuis cet écran, on a le choix d'imprimer l'inventaire, d'effacer un article ou d'éditer la fiche d'un article.

Modéliser cette situation par un diagramme de cas d'utilisation

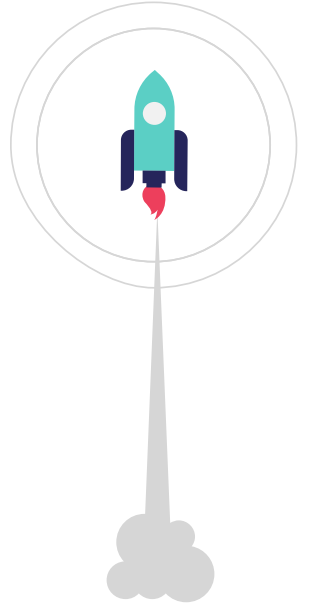
## Exercice 2:

Dans un établissement scolaire, on désire gérer la réservation des salles de cours ainsi que du matériel pédagogique (ordinateur portable ou/et Vidéo projecteur). Seuls les enseignants sont habilités à effectuer des réservations (sous réserve de disponibilité de la salle ou du matériel). Le planning des salles peut quant à lui être consulté par tout le monde (enseignants et étudiants). Par contre, le récapitulatif horaire par enseignant (calculé à partir du planning des salles) ne peut être consulté que par les enseignants. Enfin, il existe pour chaque formation un enseignant responsable qui seul peut éditer le récapitulatif horaire pour l'ensemble de la formation.

Modéliser cette situation par un diagramme de cas d'utilisation.

# **Vision statique du système :**

## **Diagramme de classes**



# Diagramme de classes

## Définition

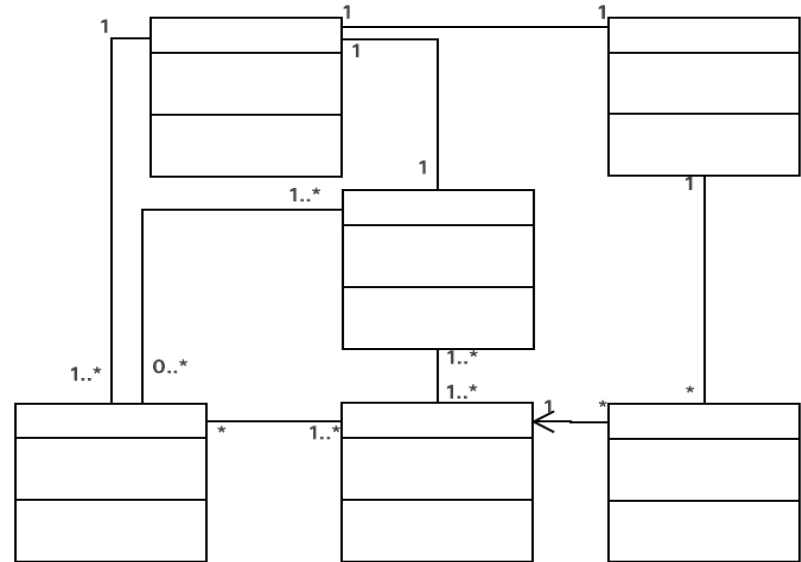
- Le diagramme de classes exprime la structure statique du système en termes de classes et de relations entre ces classes.
- L'intérêt du diagramme de classe est de modéliser les entités du système d'information.
- Le diagramme de classe permet de représenter l'ensemble des informations finalisées qui sont gérées par le domaine. Ces informations sont structurées, c'est-à-dire qu'elles ont regroupées dans des classes.



# Diagramme de classes

Le diagramme de classes comporte 5 concepts :

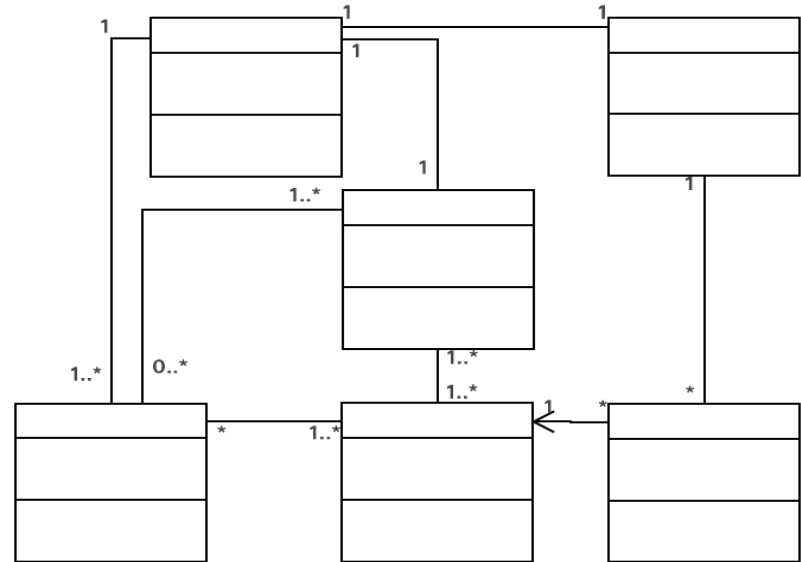
- ✓ Classe
- ✓ Attribut
- ✓ Identifiant
- ✓ Relation
- ✓ Opération



# Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ **Classe**
- ✓ Attribut
- ✓ Identifiant
- ✓ Relation
- ✓ Opération



# Classe

## Définition :

- une classe est une description d'un ensemble d'objets d'un système (un domaine d'application) : elle définit leur structure, leur comportement et leurs relations.
- Les objets peuvent être des objets concrets ou abstraits.

## Exemple:

Une université est un système. Et dans l'université, il y a des objets qui font partie de son système :

Enseignant, étudiant, classe, cours, planning, etc ...

Les objets concrets sont: Enseignant et étudiant.

Les objets abstraits sont: classe, cours, planning.

# Classe

## Représentation :

les classes sont représentées par des rectangles compartimentés :

- le 1er compartiment représente le nom de la classe
- le 2ème compartiment représente les attributs de la classe
- le 3ème compartiment représente les opérations de la classe

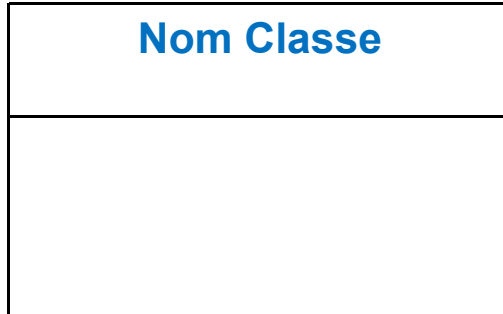
Nom Classe
Attribut 1: int Attribut 2: int Attribut 3: int
Opération 1(): void Opération 2(): void

# Classe

## Représentation :

Les compartiments d'une classe peuvent être supprimés (en totalité ou en partie) lorsque leur contenu n'est pas pertinent dans le contexte d'un diagramme. La suppression des compartiments reste purement visuelle : elle ne signifie pas qu'il n'y a pas d'attribut ou d'opération.

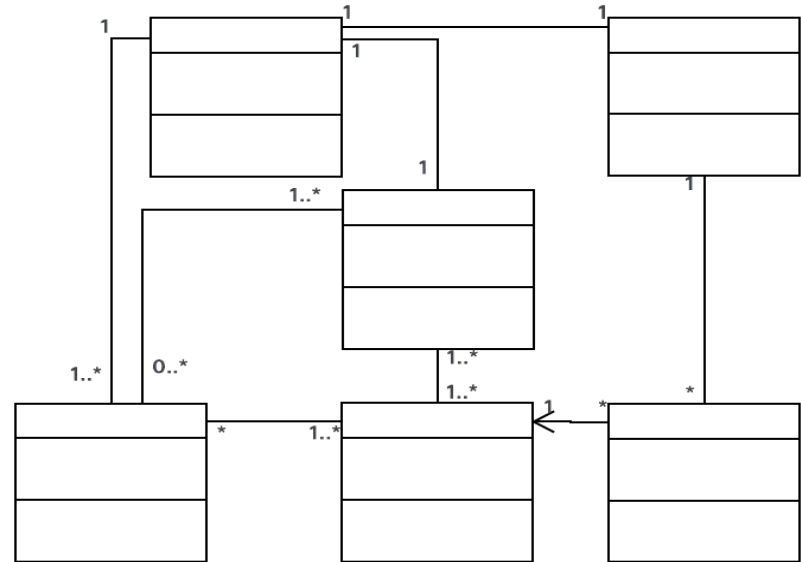
Formalisme de représentation simplifiée :



# Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ **Attribut**
- ✓ Identifiant
- ✓ Relation
- ✓ Opération



# La notion d'attribut

Une classe correspond à un concept global d'information et se compose d'un ensemble d'informations élémentaires, appelées attributs de classe.

Un attribut représente la modélisation d'une information élémentaire représentée par son nom et son format.

Nom Classe
Attribut 1: int Attribut 2: int Attribut 3: int
Opération 1(): void Opération 2(): void

# La notion d'attribut

UML définit 3 niveaux de visibilité pour les attributs :

- public (+) : l'élément est visible pour tous les clients de la classe
- protégé (#) : l'élément est visible pour les sous-classes de la classe
- privé (-) : l'élément n'est visible que par les objets de la classe dans laquelle il est déclaré.

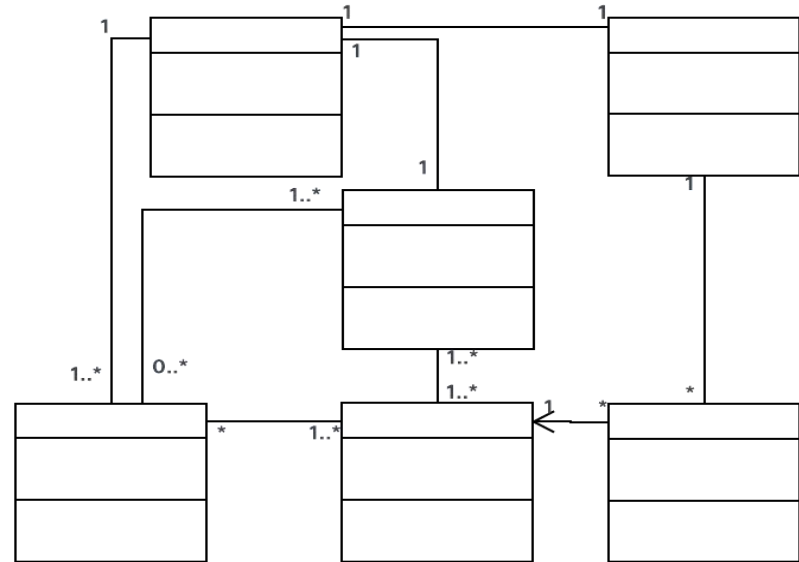
Nom Classe
+ Attribut public: int # Attribut protégé: int - Attribut privé: int



# Diagramme de classes

Le diagramme de classes comporte 5 concepts :

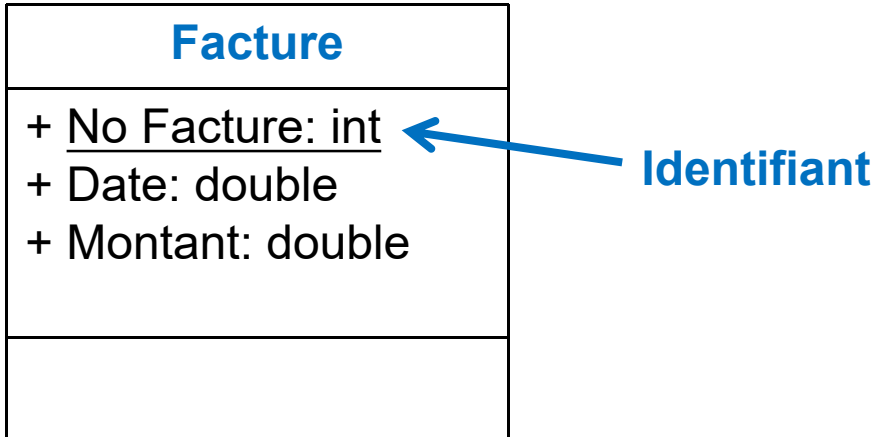
- ✓ Classe
- ✓ Attribut
- ✓ **Identifiant**
- ✓ Relation
- ✓ Opération



# La notion d'identifiant

L'identifiant est un attribut particulier, qui permet de repérer de façon unique chaque objet, instance de la classe.

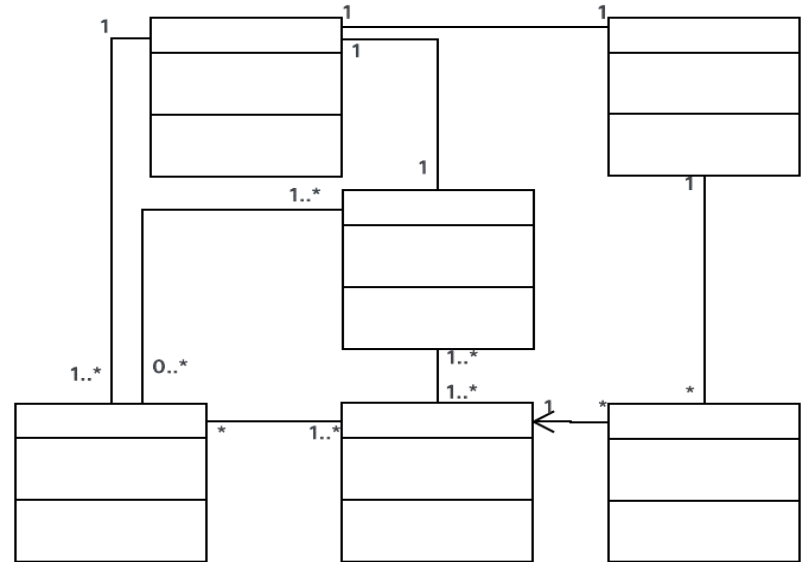
## Exemple:



# Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ Attribut
- ✓ Identifiant
- ✓ **Opération**
- ✓ Relation



# La notion d'opération

l'opération représente un élément de comportement des objets, défini de manière globale dans la classe.

Une opération est une fonctionnalité assurée par une classe. La description des opérations peut préciser les paramètres d'entrée et de sortie ainsi que les actions élémentaires à exécuter.

Facture	
+ <u>No Facture</u> : int + Date: double + Montant: double	
Editer() :	void()
Créer() :	void()
Consulter() :	void()

# La notion d'opération

Comme pour les attributs, on retrouve 3 niveaux de visibilité pour les opérations :

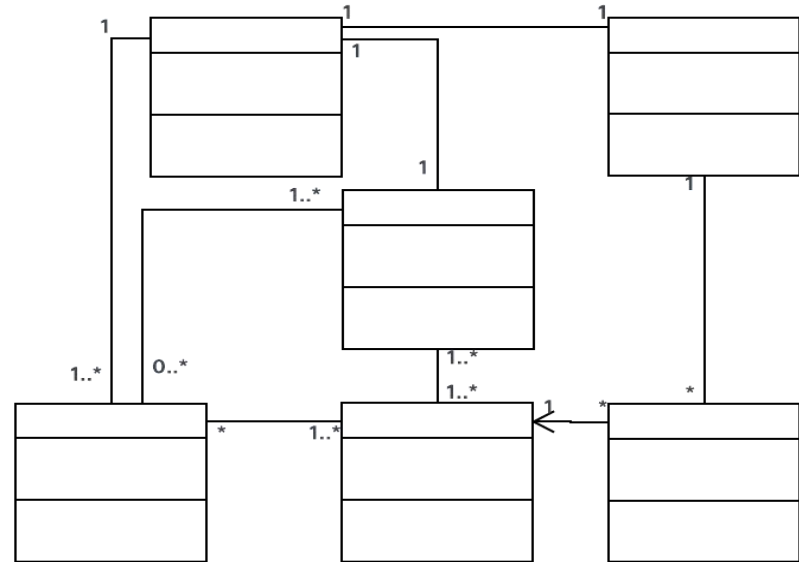
- **public (+)** : l'opération est visible pour tous les clients de la classe,
- **protégé (#)** : l'opération est visible pour les sous-classes de la classe,
- **privé (-)** : l'opération n'est visible que par les objets de la classe dans laquelle elle est déclarée.

Facture
+ No Facture: <u>int</u> + Date: double + Montant: double
+ Op public () # Op protégée () - Op privée ()

# Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ Attribut
- ✓ Identifiant
- ✓ Opération
- ✓ **Relation**



# Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

Il existe plusieurs types de relations entre classes :

- l'association
- la généralisation
- la dépendance

# Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

Il existe plusieurs types de relations entre classes :

- **l'association**
- la généralisation
- la dépendance



# Relation: Association

- L'association est la relation la plus courante et la plus riche du point de vue sémantique.
- L'association existe entre les classes et non entre les instances : elle est introduite pour montrer une structure et non pour montrer des échanges de données.
- Chaque classe qui participe à l'association joue un rôle. Les rôles sont définis par 2 propriétés :
  - la multiplicité.
  - la navigabilité.

# Relation: Association

## la multiplicité

Elle définit le nombre d'instances de l'association pour une instance de la classe. La multiplicité est définie par un nombre entier ou un intervalle de valeurs.

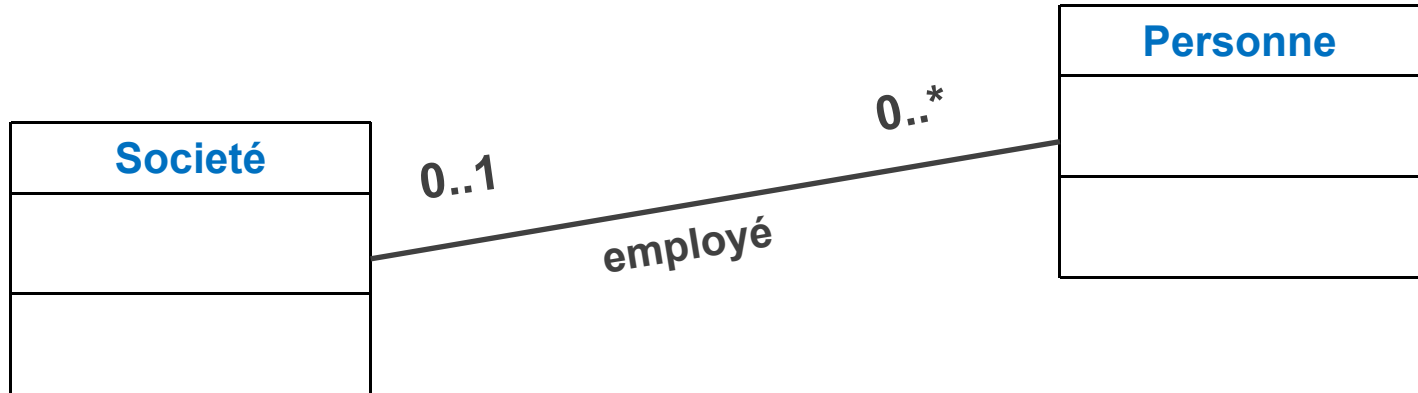
La multiplicité est notée sur le rôle (elle est notée à l'envers de la notation MERISE).

1	Un et un seul
0..1	Zéro ou un
N ou *	N (entier naturel)
N..M	De M à N (entiers naturels)
0..*	De zéro à plusieurs
1..*	De 1 à plusieurs

# Relation: Association

la multiplicité

Formalisme et exemple en employant le noms de l'association et la multiplicité des rôles :



# Relation: Association

## la multiplicité

La multiplicité peut également s'exprimer par des symboles :

1 

0..1 

\* 

0..\* 

1..\* 

# Relation: Association

## la navigabilité

- La navigabilité n'a rien à voir avec le sens de lecture de l'association.
- Une navigabilité placée sur une terminaison cible indique si ce rôle est accessible à partir de la source.
- On représente graphiquement la navigabilité par une flèche du côté de la terminaison navigable et on empêche la navigabilité par une croix du côté de la terminaison non navigable.



# Relation: Association

la navigabilité

Exemple:



- La terminaison du côté de la classe **Commande** n'est pas navigable : cela signifie que les instances de la classe **Produit** ne stockent pas de liste d'objets du type **Commande**.
- La terminaison du côté de la classe **Produit** est navigable : chaque objet commande contient une liste de produits.

# Relation: Association

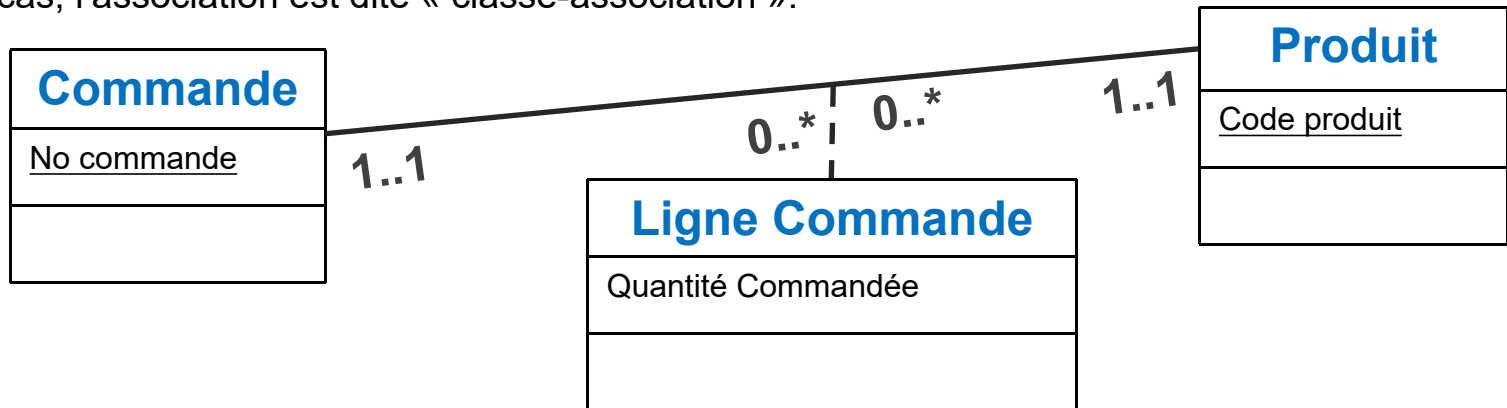
## Les classes-association:

Les attributs d'une classe dépendent fonctionnellement de l'identifiant de la classe. Parfois, un attribut dépend fonctionnellement de 2 identifiants, appartenant à 2 classes différentes.

### Exemple:

L'attribut « quantité commandée » dépend fonctionnellement du numéro de commande et du code produit. On va donc placer l'attribut « quantité commandée » dans l'association « commander ».

Dans ce cas, l'association est dite « classe-association ».

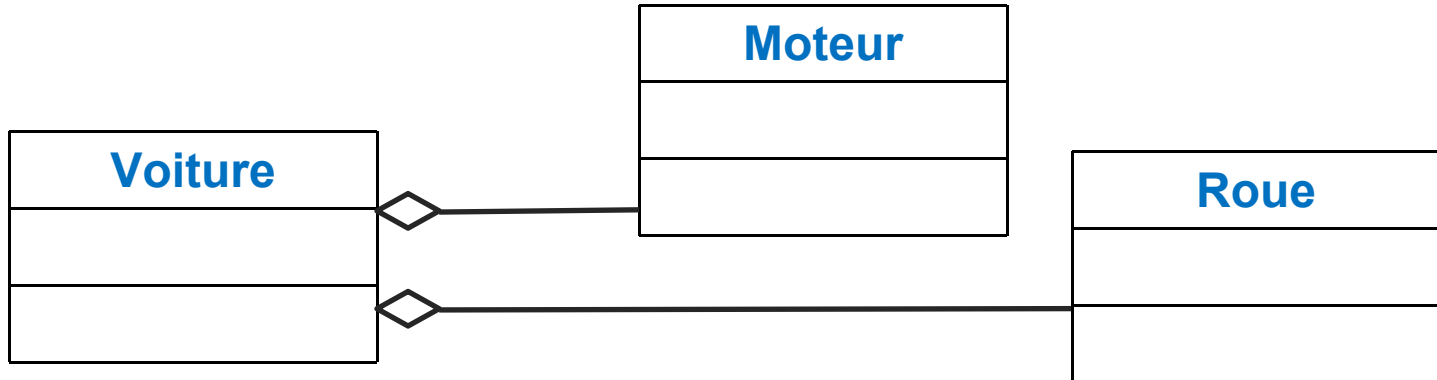


# Relation: Association

## L'agrégation:

- Une agrégation représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité.
- L'agrégation se représente toujours avec un petit losange du côté de l'agregat.

## Formalisme et exemple :



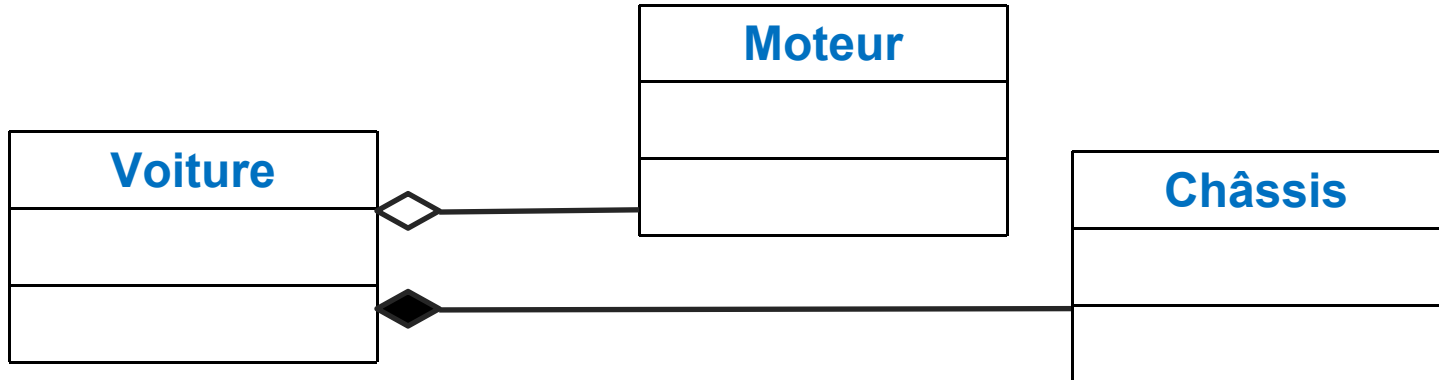


# Relation: Association

## La composition:

- Il s'agit d'une appartenance forte. La vie de l'objet composant est liée a celle de son composé.
- La composition se modélise par un losange noir côté composé.

## Formalisme et exemple :



# Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

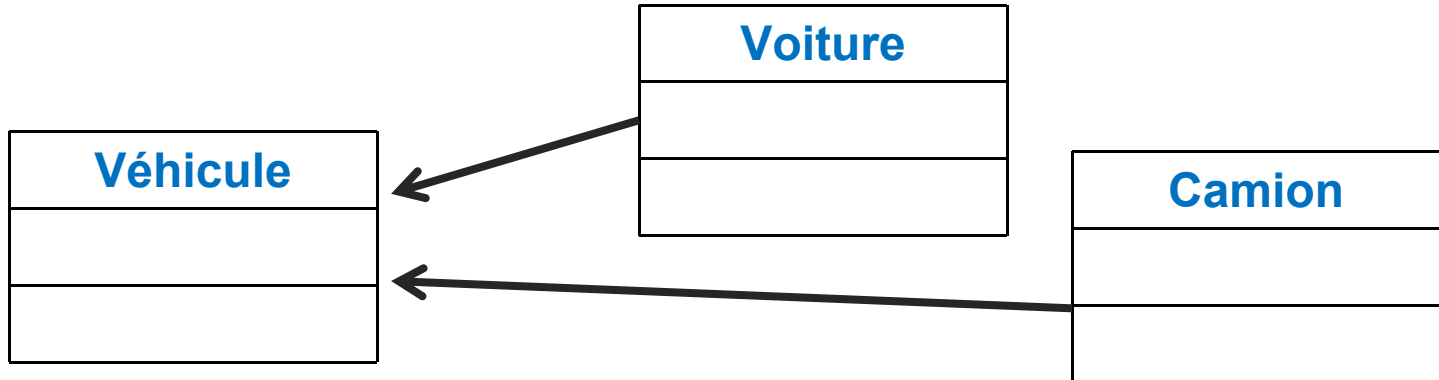
Il existe plusieurs types de relations entre classes :

- l'association
- **la généralisation**
- la dépendance

# Relation: Généralisation

Une relation de généralisation est une relation dans laquelle un élément de modèle (l'enfant) est basé sur un autre élément de modèle (le parent).

**Formalisme et exemple :**



# Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

Il existe plusieurs types de relations entre classes :

- l'association
- la généralisation
- **la dépendance**

# Relation: Dépendance

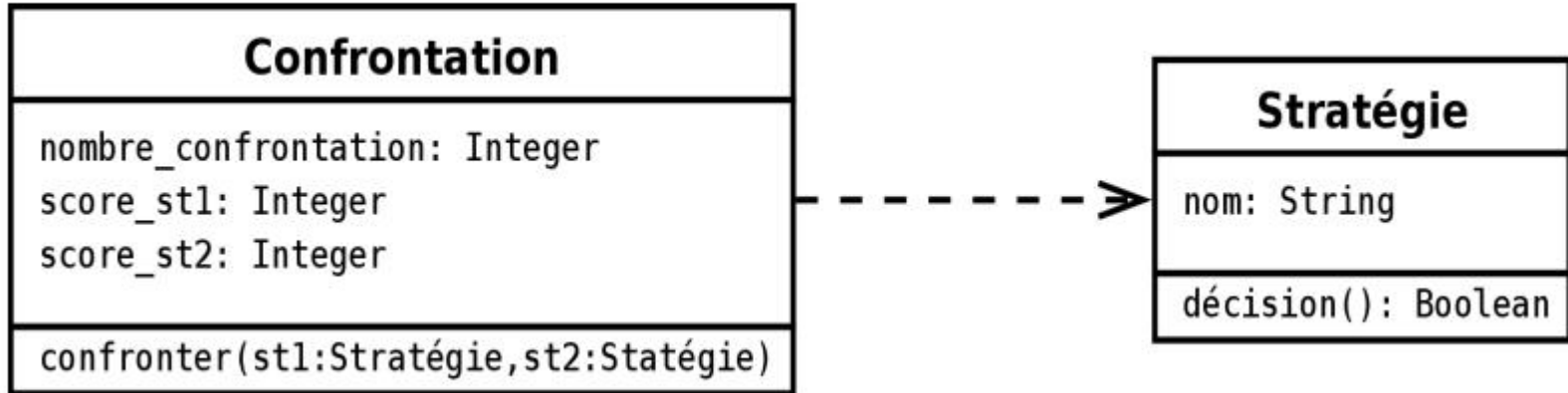
- Une dépendance est une relation entre deux éléments dans laquelle une modification à un élément provoque des modifications sur l'autre élément .
- Une dépendance est représentée graphiquement sous la forme d'une ligne dirigée de pointillés, dirigée vers l'élément.



# Relation: Dépendance

## Exemple :

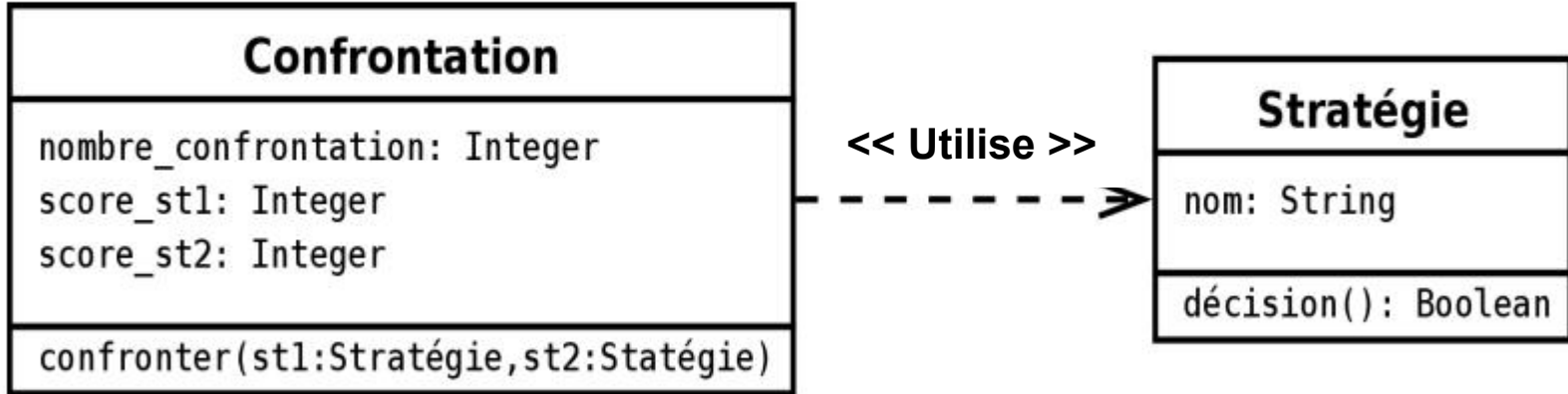
une modification à la stratégie (l'élément indépendant) provoque des modifications sur la confrontation (l'élément dépendant).



# Relation: Dépendance

- La dépendance peut être stéréotypée pour préciser le lien sémantique entre les éléments du modèle.

## Exemple :



# **Diagramme de classes**

**Exemple:**

**Une application pour la gestion d'une  
bibliothèque**



# Diagramme de classes

## Étape 1 : Identifier les noms de classes

La première étape consiste à identifier les principaux objets du système.

<b>Utilisateur</b>

<b>Commande</b>

<b>Livre</b>

<b>roman</b>

<b>Livre scientifique</b>

<b>Auteur</b>

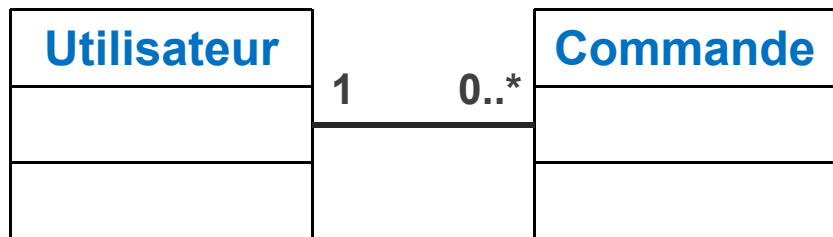
<b>Résumé</b>

<b>livres pour enfants</b>

<b>livres de recettes</b>

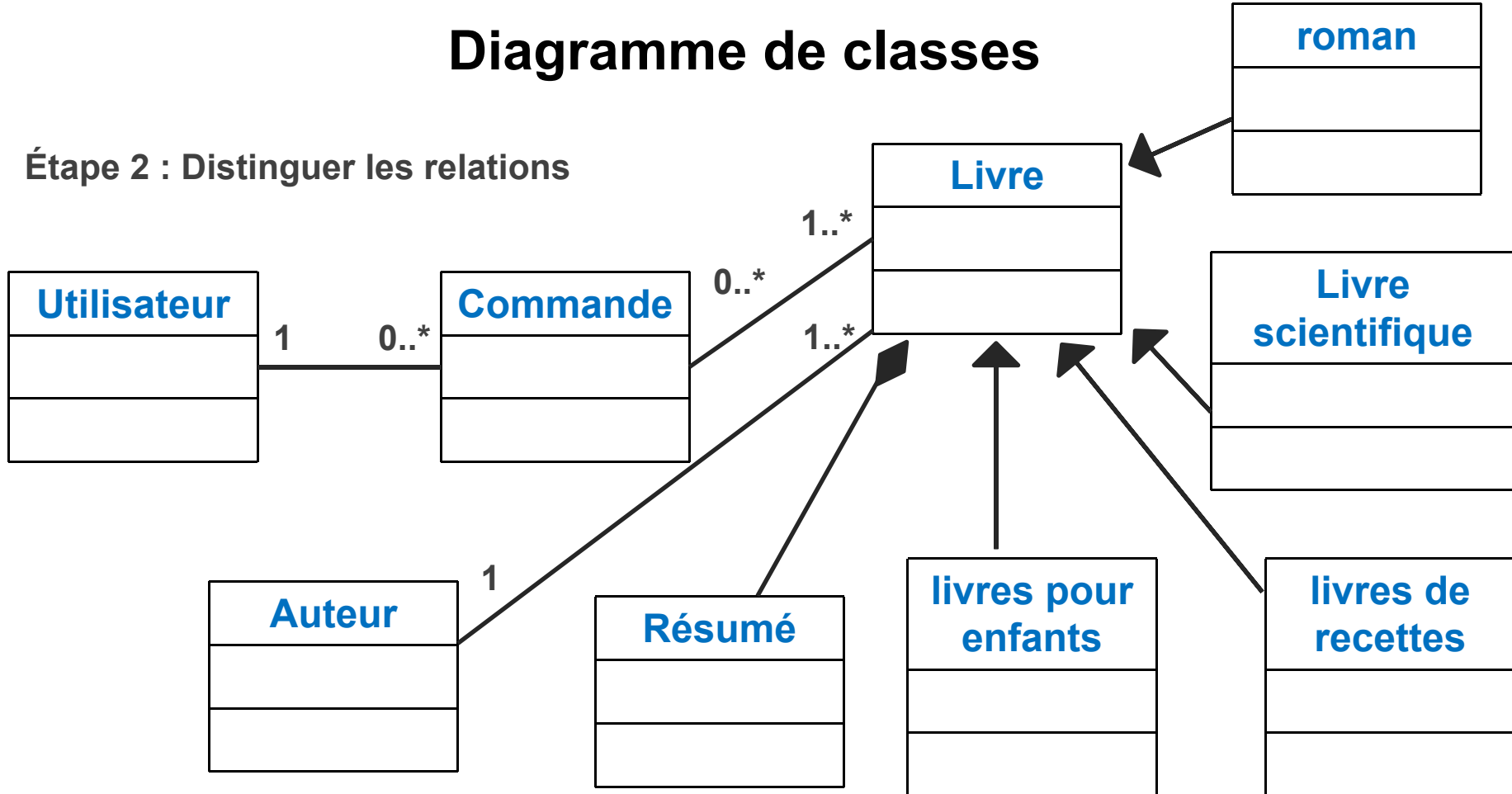
# Diagramme de classes

Étape 2 : Distinguer les relations



# Diagramme de classes

Étape 2 : Distinguer les relations



# Diagramme de classes

## Étape 3 : Créer la structure

Ajouter des attributs et des fonctions/méthodes/opérations.

# Diagramme de classes

**Exercices**

## Gestion d'un hôtel

Un hôtel est composé d'au moins deux chambres. Chaque chambre dispose d'une salle d'eau : douche ou bien baignoire. Un hôtel héberge des personnes. Il peut employer du personnel et il est impérativement dirigé par un directeur. On ne connaît que le nom et le prénom des employés, des directeurs et des occupants. Certaines personnes sont des enfants et d'autres des adultes (faire travailler des enfants est interdit). Un hôtel a les caractéristiques suivantes : une adresse, un nombre de pièces et une catégorie.

Une chambre est caractérisée par le nombre et de lits qu'elle contient, son prix et son numéro. On veut pouvoir savoir qui occupe quelle chambre à quelle date. Pour chaque jour de l'année, on veut pouvoir calculer le loyer de chaque chambre en fonction de son prix et de son occupation (le loyer est nul si la chambre est inoccupée). La somme de ces loyers permet de calculer le chiffre d'affaires de l'hôtel entre deux dates.

## Gestion d'un hôtel

Un hôtel est composé d'au moins deux **chambres**. Chaque chambre dispose d'une **salle d'eau** : **douche** ou bien **baignoire**. Un hôtel héberge des **personnes**. Il peut **employer du personnel** et il est impérativement dirigé par **un directeur**. On ne connaît que le nom et le prénom des employés, des directeurs et des occupants. Certaines personnes sont des **enfants** et d'autres des **adultes** (faire travailler des enfants est interdit). Un hôtel a les caractéristiques suivantes : une adresse, un nombre de pièces et une catégorie.

Une chambre est caractérisée par le nombre et de lits qu'elle contient, son prix et son numéro.

**On veut pouvoir savoir qui occupe quelle chambre à quelle date.** Pour chaque jour de l'année, on veut pouvoir calculer le loyer de chaque chambre en fonction de son prix et de son occupation (le loyer est nul si la chambre est inoccupée). La somme de ces loyers permet de calculer le chiffre d'affaires de l'hôtel entre deux dates.

## **Gestion des défauts pour un fabricant de moteurs**

Une usine de fabrication des moteurs cherche à organiser la gestion des défauts observés sur des moteurs confrontés à des tests en situation réelle.

Pour chaque moteur on relève tous les défauts observés à des dates différentes. Pour chacun de ces défauts on note le kilométrage et on diagnostique la gravité.

Les défauts sont classés par type, chaque type de défaut étant codifié (on peut imaginer par exemple, la casse, la fuite, etc.).

Les défauts sont relevés par des opérateurs, employés du groupe et affectés à une usine.

Les moteurs sont d'un modèle particulier, fabriqué dans une usine, et sous la responsabilité d'un responsable également employé du groupe.



## Gestion des défauts pour un fabricant de moteurs

Une usine de fabrication des moteurs cherche à organiser la gestion des défauts observés sur des moteurs confrontés à des tests en situation réelle.

Pour chaque moteur on relève tous les défauts observés à des dates différentes. Pour chacun de ces défauts on note le kilométrage et on diagnostique la gravité.

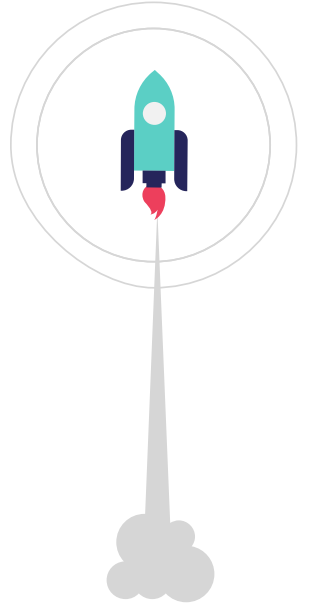
Les défauts sont classés par type, chaque type de défaut étant codifié (on peut imaginer par exemple, la casse, la fuite, etc.).

Les défauts sont relevés par des opérateurs, employés du groupe et affectés à une usine.

Les moteurs sont d'un modèle particulier, fabriqué dans une usine, et sous la responsabilité d'un responsable également employé du groupe.

# **Vision statique du système :**

## **Diagramme d'objets**



# Diagramme d'objets

## Définition

- Un diagramme d'objets est une instance d'un diagramme de classes : il représente des objets (instances de classes) et leurs liens (instances de relations).
- Un diagramme d'objets donne une vue figée de l'état d'un système à un instant donné.

# Diagramme d'objets

## Définition

- Les diagrammes d'objets s'utilisent principalement :
  - illustrer le modèle de classes en montrant un exemple qui explique le modèle ;
  - préciser certains aspects du système en mettant en évidence des détails imperceptibles dans le diagramme de classes;
  - exprimer une exception en modélisant des cas particuliers ou des connaissances non généralisables qui ne sont pas modélisés dans un diagramme de classe ;
  - faciliter la compréhension des structures de données complexes, structures récursives.
- **Le diagramme de classes modélise les règles et le diagramme d'objets modélise des faits.**

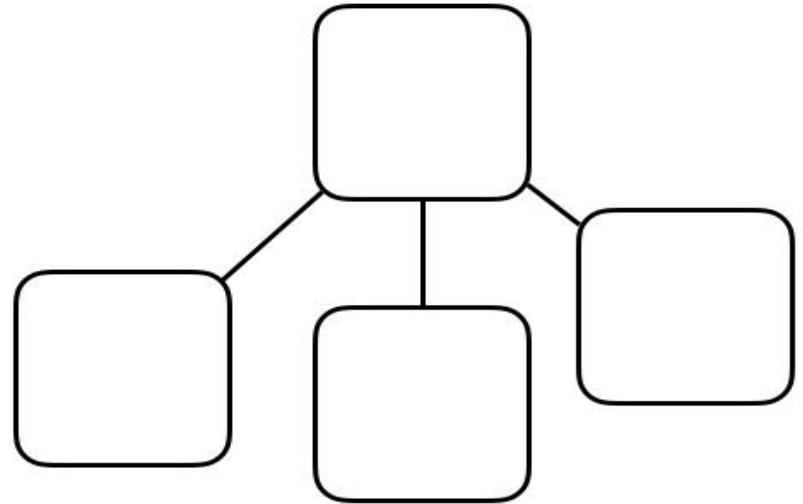
# Diagramme d'objets

Le diagramme de classes comporte 2

concepts :

- ✓ Les objets (instances de classes)
- ✓ Les liens (instances d'associations)

**«La notation des diagrammes d'objets est  
dérivée de celle des diagrammes de classes»**



# La notion d'objets

- Un objet est une instance d'une classe : il représente "l'état" d'une classe à un instant précis.
- Représentation UML:

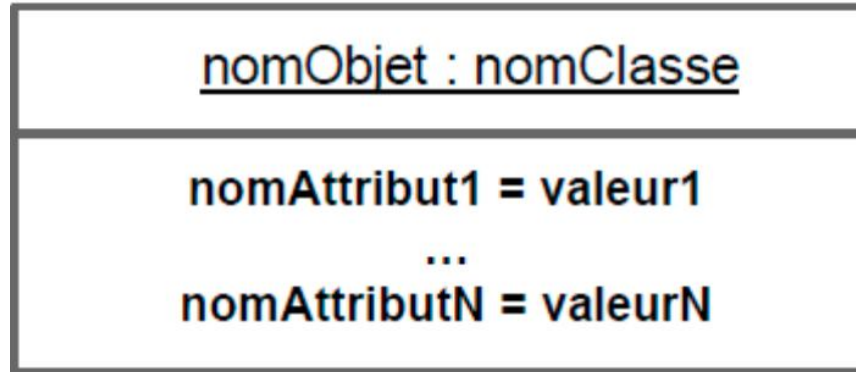
Nom de l'objet

Nom de l'objet: NomClasse

: NomClasse

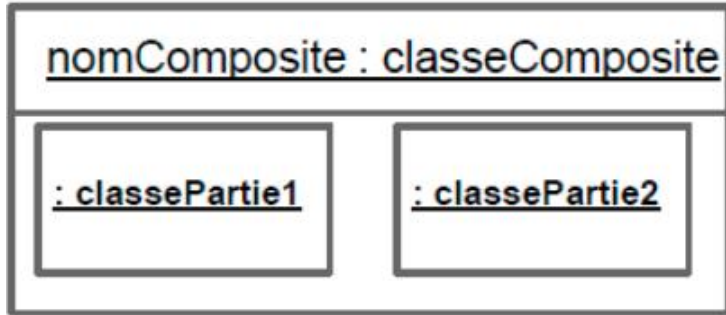
# La notion d'objets

- L'état d'un objet est déterminé par les valeurs de ses attributs : Les représentations des objets peuvent contenir des attributs significatifs.
- Les valeurs des attributs sont optionnelles.

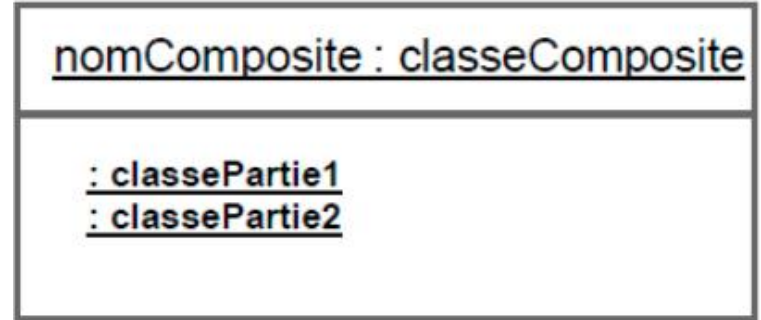


# La notion d'objets composites

- Un objet composé de sous-objets est appelée composite.
- Les objets composites sont issus de classes liées par une composition.
- Les attributs d'un objet composite sont eux-même des objets.
- Les sous-objets peuvent être représentés de manière graphique ou textuelle.



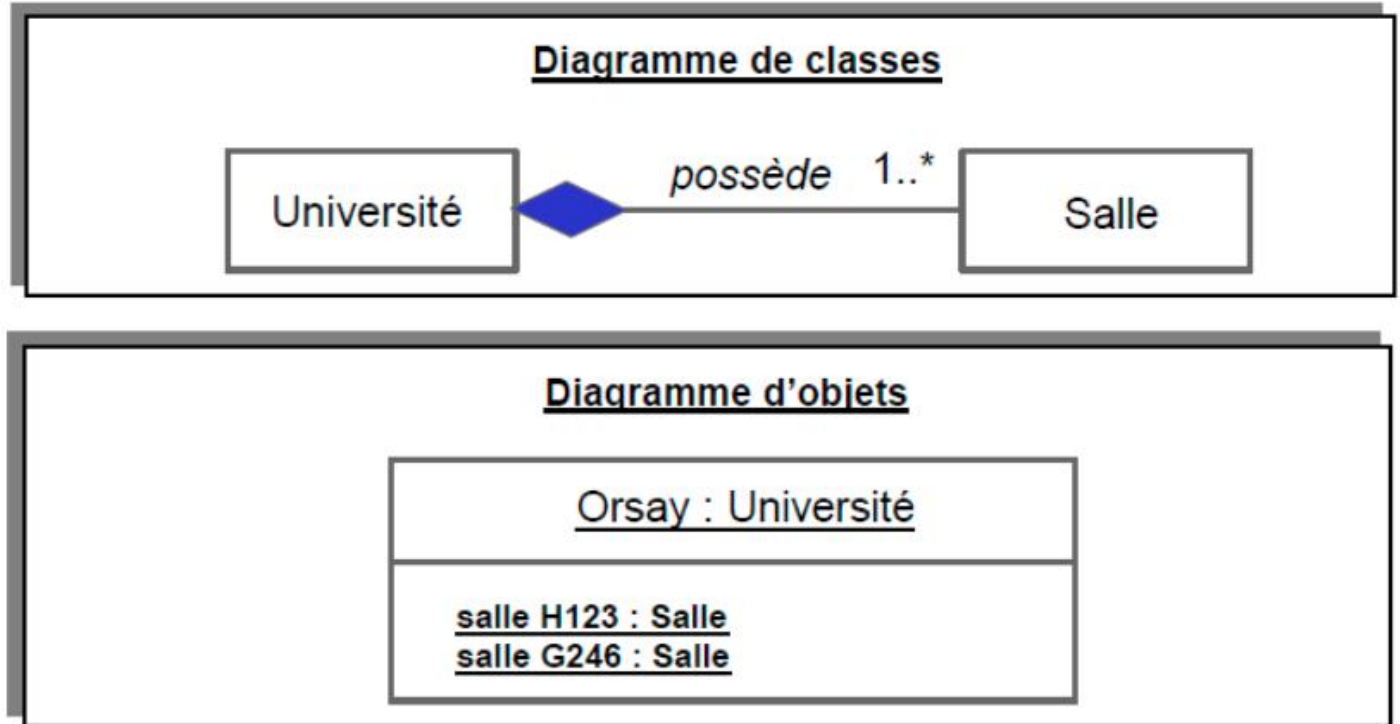
OU





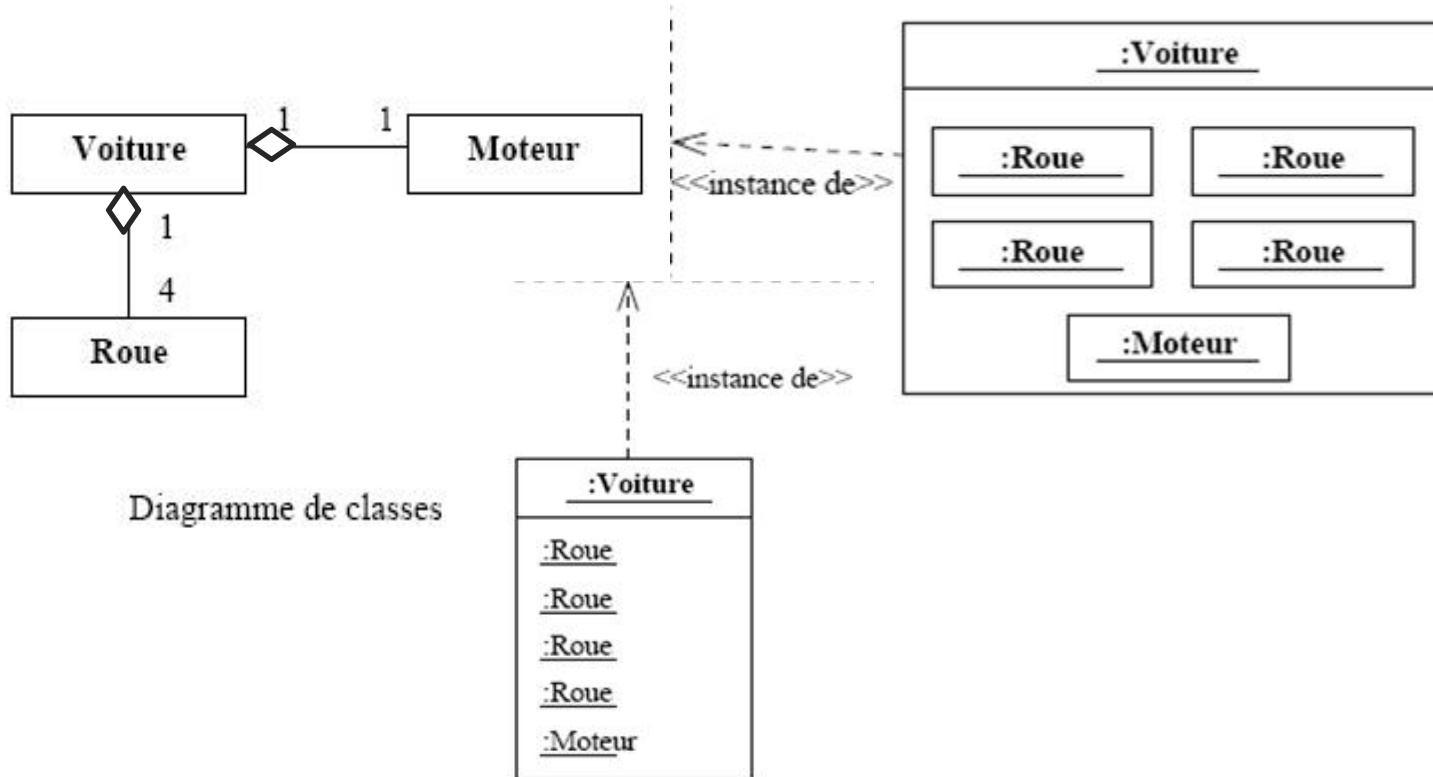
# La notion d'objets composites

## Exemple: association de composition



# La notion d'objets composites

## Exemple: association d'agrégation



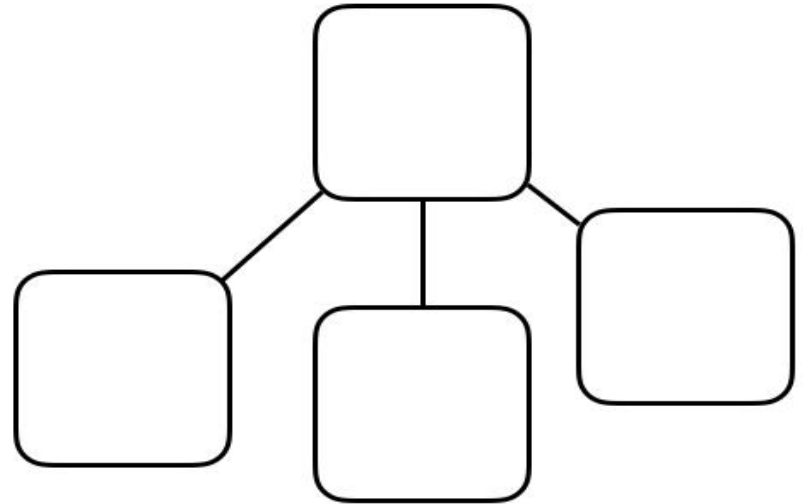
# Diagramme d'objets

Le diagramme de classes comporte 2

concepts :

- ✓ Les objets (instances de classes)
- ✓ Les liens (instances d'associations)

**«La notation des diagrammes d'objets est  
dérivée de celle des diagrammes de classes»**



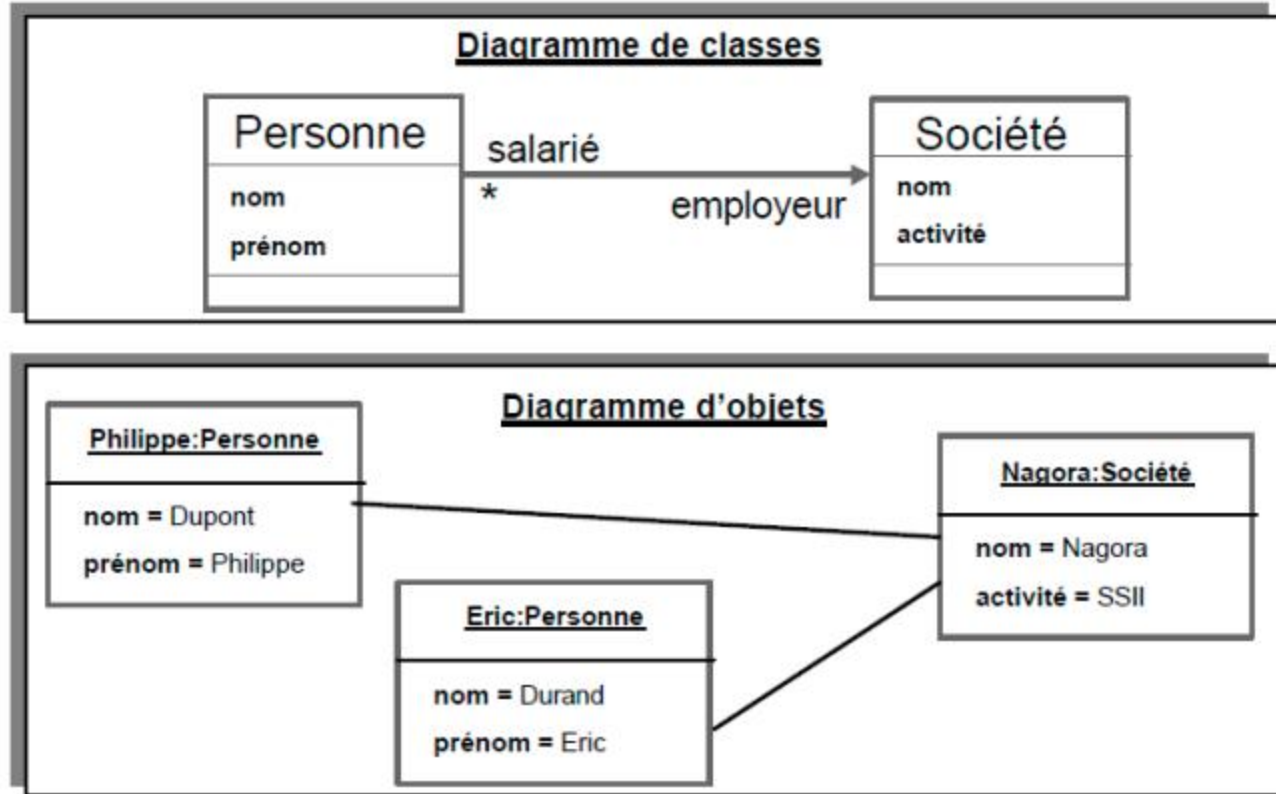
# La notion des liens

- Les objets sont reliés par des instances d'associations : les liens.
- Un lien représente une relation entre objets à un instant donné.

**ATTENTION : la multiplicité des extrémités des liens est toujours de 1.**

# La notion des liens

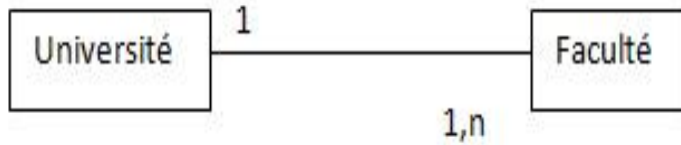
## Exemple:



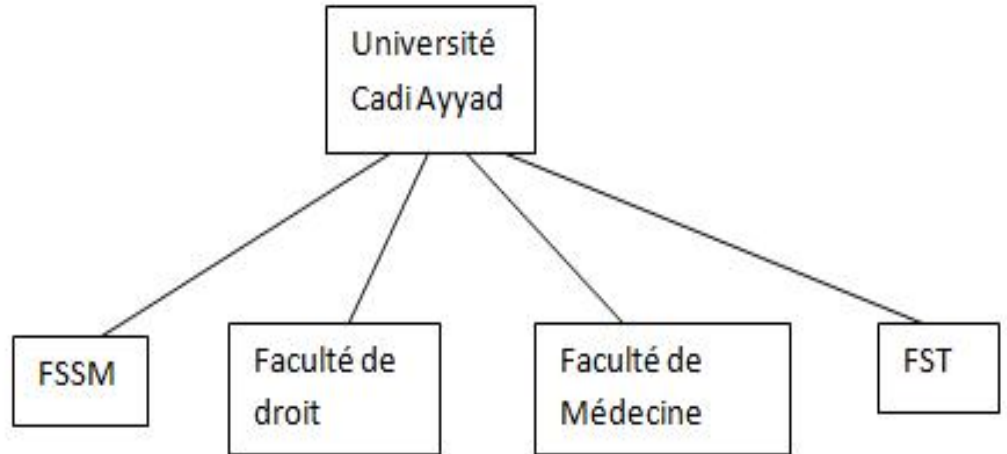
# La notion des liens

## Exemple:

### Diagramme de classes:



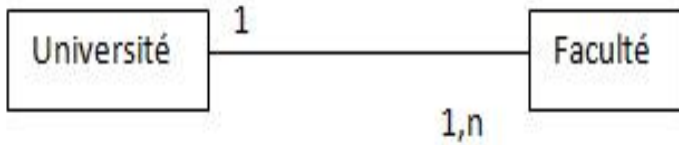
### Diagramme d'objets:



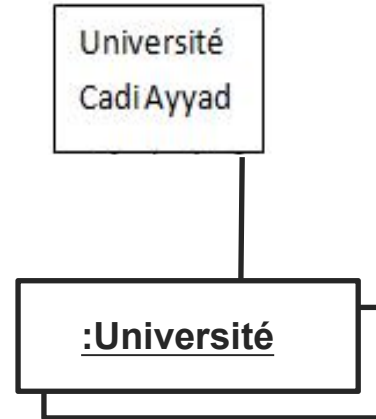
# La notion des liens

## Exemple:

### Diagramme de classes:



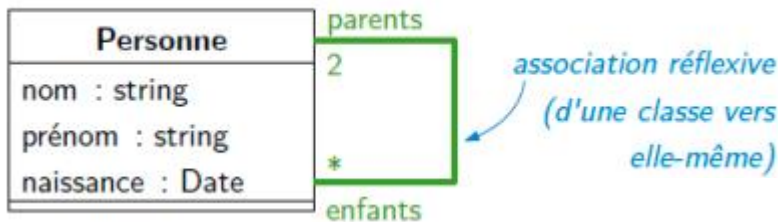
### Diagramme d'objets:



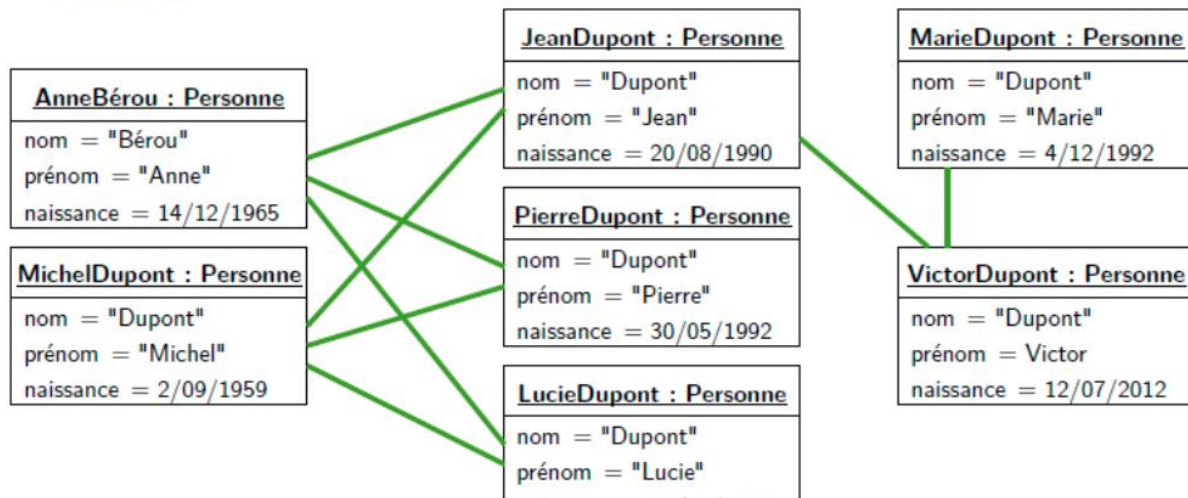
# La notion des liens: relation réflexive

- Les liens instances des associations réflexives peuvent relier un objet à lui même.

## Diagramme de classes:



## Diagramme d'objets:





# La notion des liens: relation multiple

Diagramme de classes:

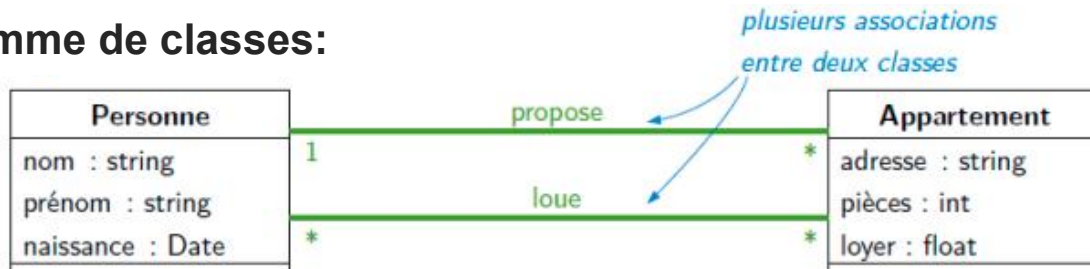
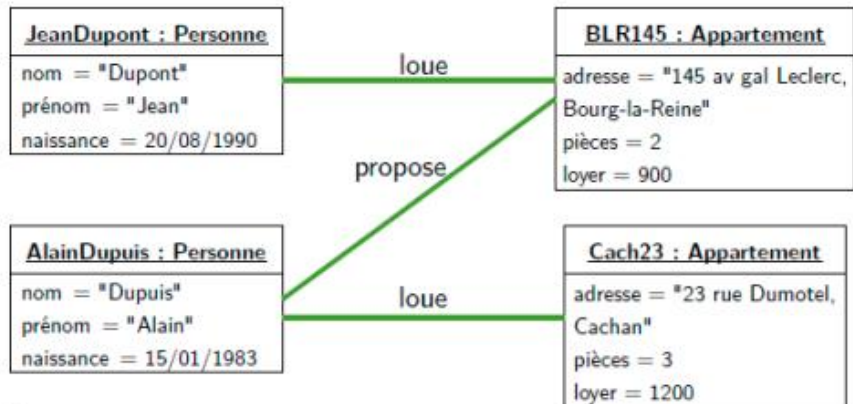


Diagramme d'objets:



# La notion des liens: relation de la généralisation

- La relation de généralisation ne possède pas d'instance, elle n'est donc jamais représentée dans un diagramme d'objets.
- Une classe mère est instanciée à travers ses classes filles.

# Diagramme d'objets

## Exercices

# Exercice 1

Un objet graphique nommé "validation" de la classe Bouton et en état "active" est en relation avec PNG133, un objet graphique de type Image.

Un ensemble d'autre boutons anonymes dont l'état est "désactivé" sont aussi liés à PNG133.

L'image peut modifier sa taille grâce à un objet glisse qui est nommé: G9.

➤ Dessinez le diagramme d'objets correspondant à la situation décrite ci-dessus.

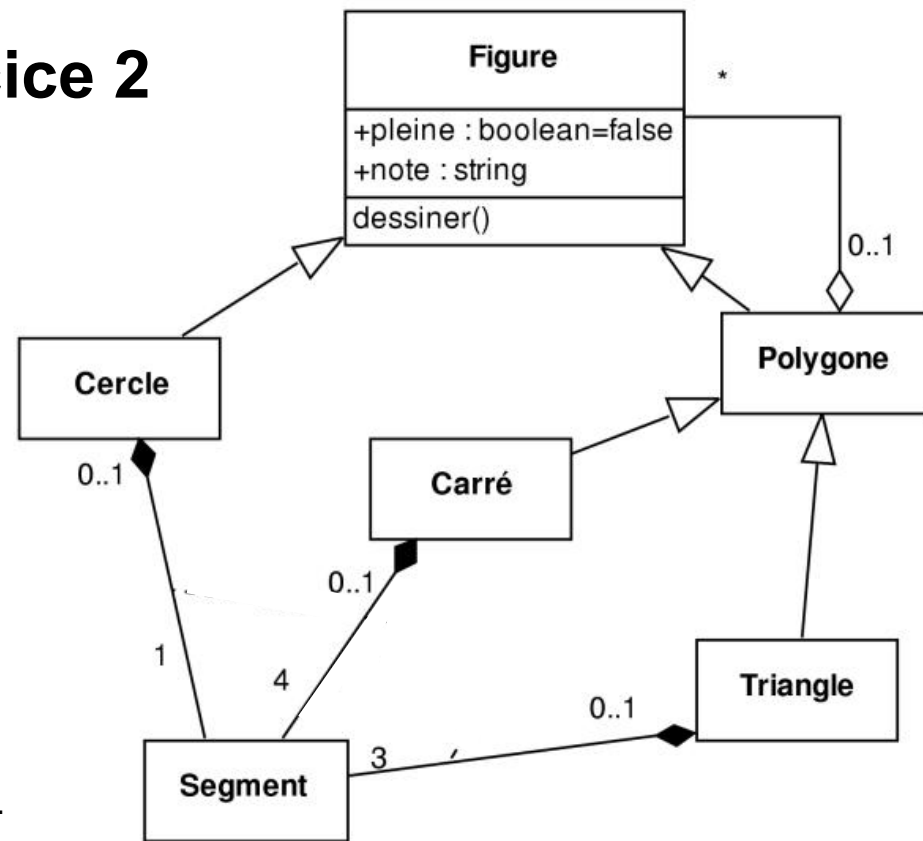
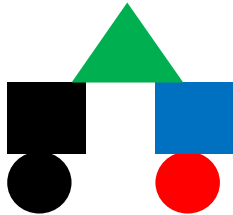
## Exercice 2

Considérez le diagramme de classes suivant:

Réaliser le diagramme d'objets en considération des informations suivantes:

- Cercle 1: noire, rayon=3 cm, diamètre= 9 cm.
- Cercle 2: rouge, rayon=3 cm, diamètre= 9 cm.
- Carré 1: noire, angle= 3 cm.
- Carré 2: bleu, angle= 3 cm.
- Triangle 1: vert, somme des angles= 180°.

Pour la construction de la forme ci-dessous, le carré 1 est liée au le cercle 1 et le carré 2 est liée au le cercle 2.



Un triathlète utilise trois types de moyens de déplacement : la nage, le cyclisme et la course pied. La nage consiste à nager une distance courte avec un maillot de bain dans un liquide (lac ou mer). Le cyclisme consiste à pédaler sur une distance longue avec un vélo sur une route. La course a pied consiste à courir une distance moyenne avec des chaussures sur une route.

Autrement dit, le triathlète possède des équipements (vélo, maillot ou chaussure) pour effectuer une distance (courte distance, moyenne distance ou longue distance) sur un site (liquide ou route) en utilisant un moyen de déplacement (nage, cyclisme ou course à pied).

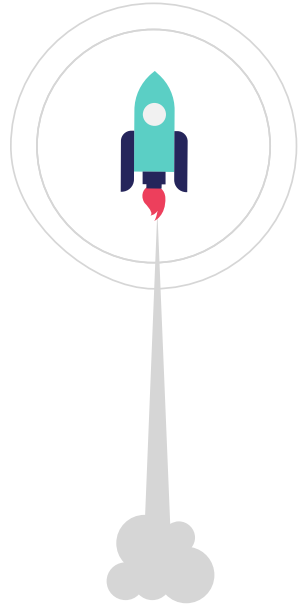
1- Dessiner un diagramme de classes.

2- Dessinez un diagramme d'objets correspondant au texte suivant :

- ☐ Ahmed est un triathlète qui court à pied une moyenne distance sur la route départementale 3 avec ses chaussures.
- ☐ Karim est un triathlète qui nage une courte distance dans la mer méditerranée avec un maillot de bain.

# **Vision statique du système :**

## **Diagramme de composants**



# Diagramme de composants

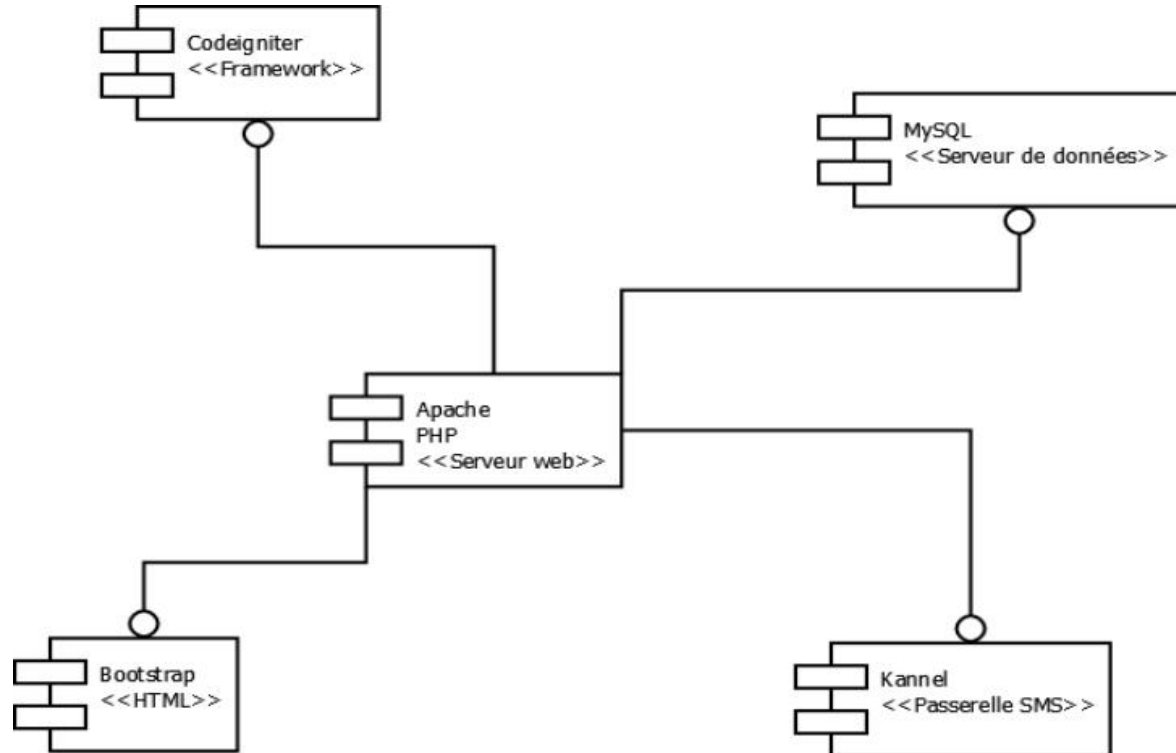
## Définition

- Une Description des composants logiciels et de leurs dépendances.
- Composant : un fichier de programme source, une bibliothèque, un programme exécutable, SGBD ...
- Utilisé en conception de logiciel pour allouer les classes et objets aux composants.
- En général, les diagrammes de composants ne sont utilisés que pour des systèmes complexes.



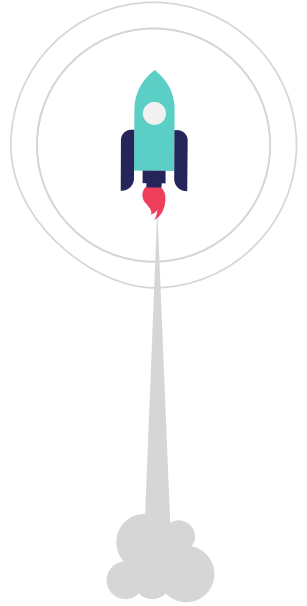
# Diagramme de composants

Exemple:



# **Vision statique du système :**

## **Diagramme de déploiement**



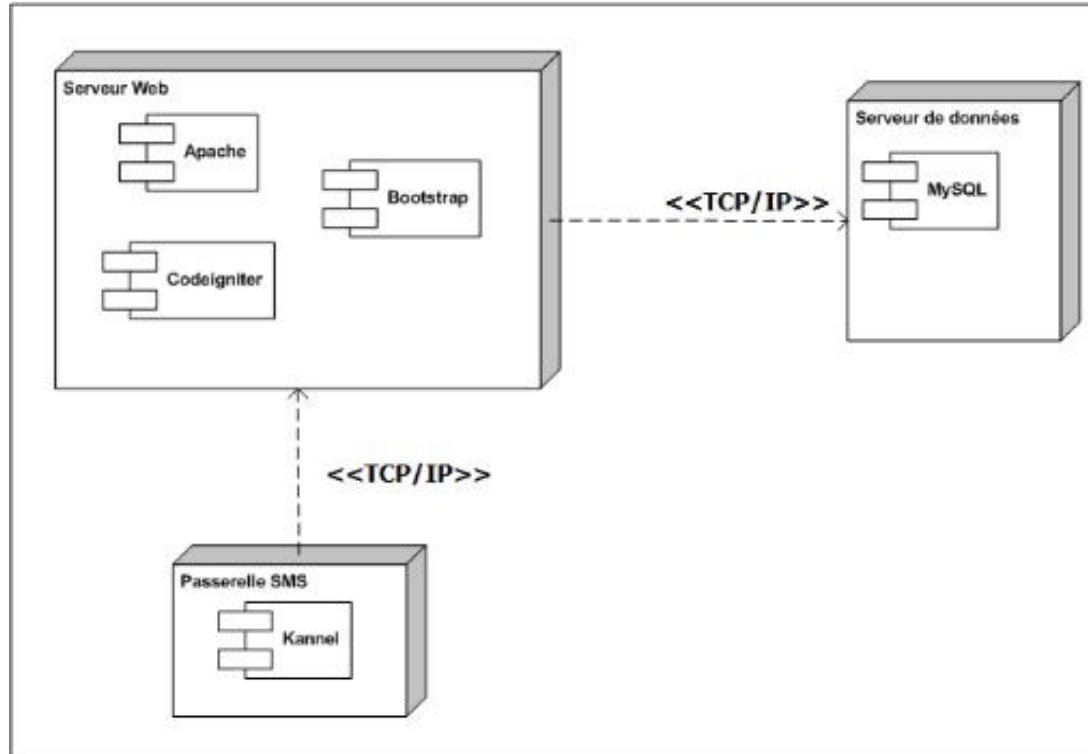
# Diagramme de déploiement

## Définition

Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels et les ressources matérielles sont représentées sous forme de noeuds. Les noeuds sont connectés entre eux, à l'aide d'un support de communication. La nature des lignes de communication et leurs caractéristiques peuvent être précisées.

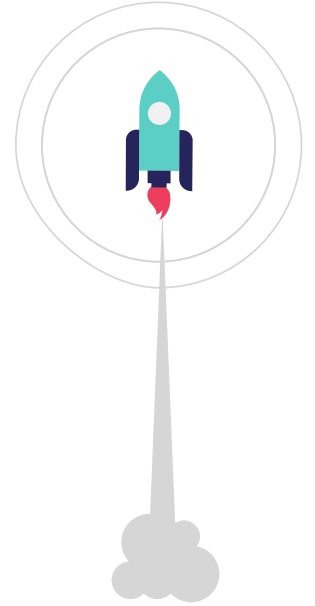
# Diagramme de déploiement

Exemple:



# **Vision Dynamique du système :**

## **Diagramme de Collaboration**



# Diagramme de collaboration

## Définition

- ❑ Extension des diagrammes d'objets(vue dynamique) : la notation des diagrammes de collaboration est dérivée de la notation des diagrammes d'objets.
- ❑ Décrit le comportement collectif d'un ensemble d'objets, en vue de réaliser une opération.
- ❑ Met en évidence les interactions entre les objets en les modélisant par des envois de messages.
- ❑ Une interaction définit la communication entre les objets sous la forme d'un ensemble partiellement ordonné de messages.

# Diagramme de collaboration

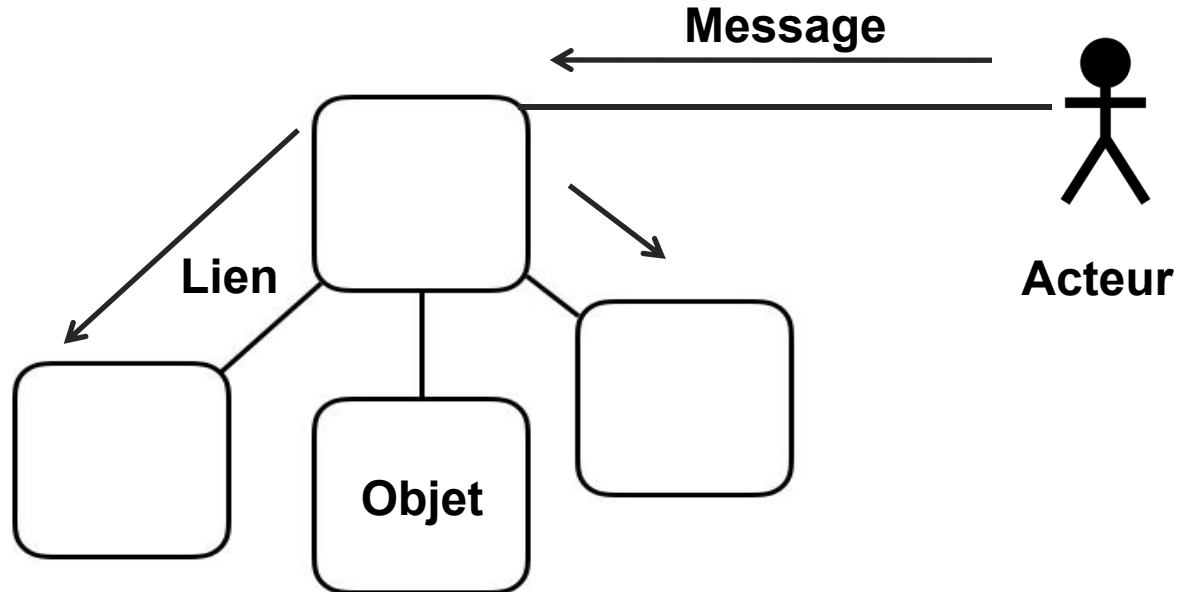
## Définition

**Un diagramme de collaboration =**

**Des Objets + Des Messages**

# Diagramme de collaboration

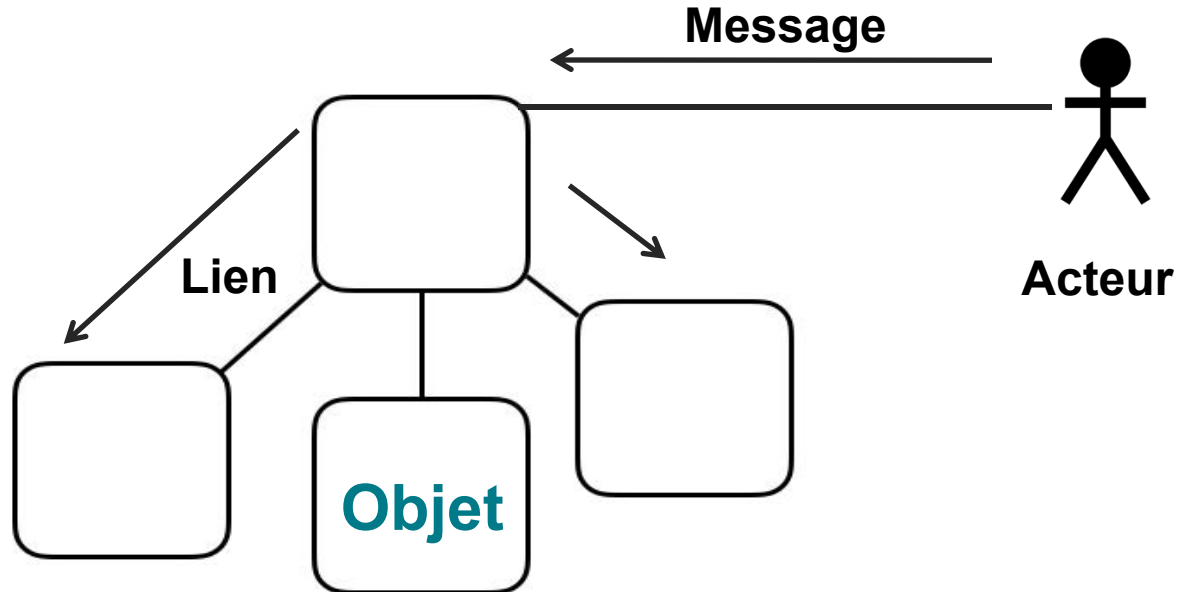
Le diagramme de collaboration comporte 4 concepts :





# Diagramme de collaboration

Le diagramme de collaboration comporte 4 concepts :



# Diagramme de collaboration

## Objet

- ❑ Description d'un objet du monde réel (instance de classe). Il peut être une personne ou une chose.
- ❑ Dans un diagramme de collaboration les objets qui présentent les personnes (Acteurs) sont des acteurs internes.
- ❑ Les acteurs externes ne doivent pas être présentés comme des objets.

# Diagramme de collaboration

## Objet

- ❑ Le formalisme d'un objet dans le diagramme de collaboration est celle du diagramme d'objets:

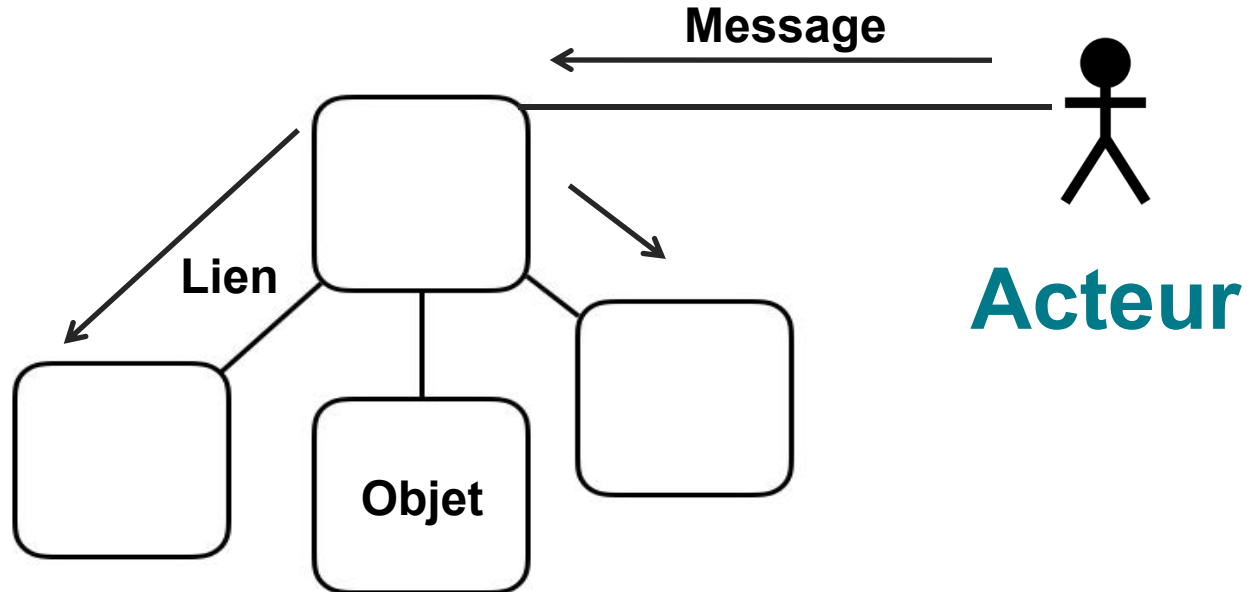
Nom de l'objet

Nom de l'objet: NomClasse

: NomClasse

# Diagramme de collaboration

Le diagramme de collaboration comporte 4 concepts :



# Diagramme de collaboration

## Acteur

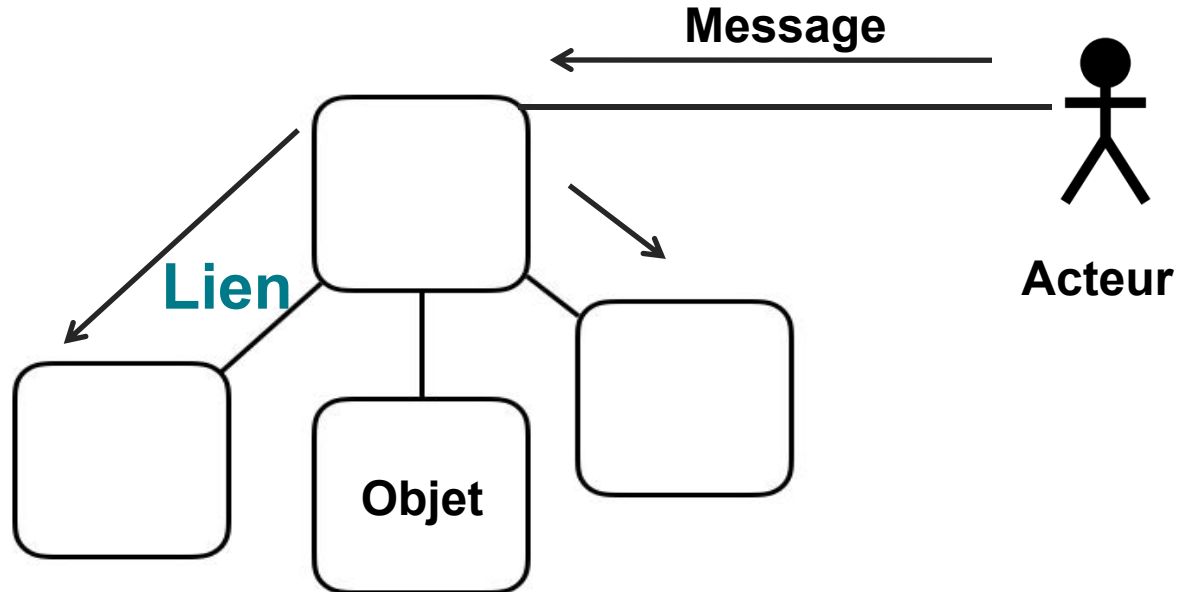
- ❑ La définition d'un acteur dans le contexte d'un diagramme de collaboration est la même que celle en vigueur dans le diagramme de cas d'utilisation; elle spécifie un utilisateur externe, ou un ensemble d'utilisateurs liés qui interagissent avec un système.
- ❑ Le formalisme d'un acteur est comme suivant:



**Nom Acteur**

# Diagramme de collaboration

Le diagramme de collaboration comporte 4 concepts :



# Diagramme de collaboration

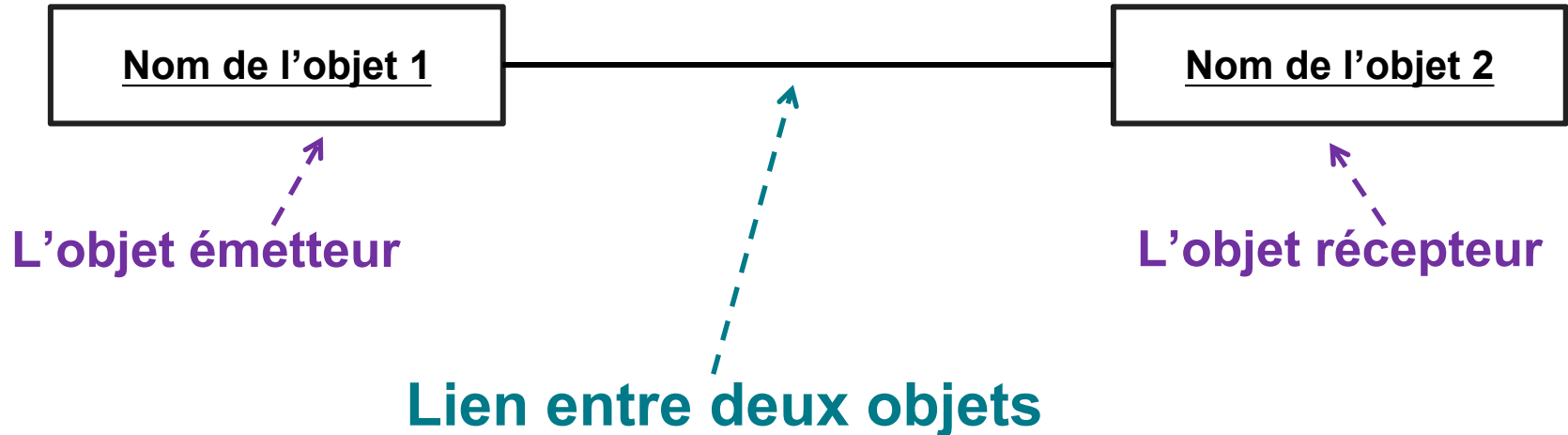
## Lien

- ❑ Un lien entre objets représente une connexion entre deux objets, il permet de mettre en exergue la collaboration entre objets, d'où le nom de diagramme de collaboration.
- ❑ Il est représenté sous forme d'un trait plein entre :
  - ❑ Deux objets
  - ❑ Un objet et un acteur (ou vice-versa)
- ❑ Un lien entre objets peut être une instance d'association entre classes (diagramme de classes).

# Diagramme de collaboration

## Objet

- ❑ Le formalisme d'un lien dans le diagramme de collaboration est comme suivant:





# Diagramme de collaboration

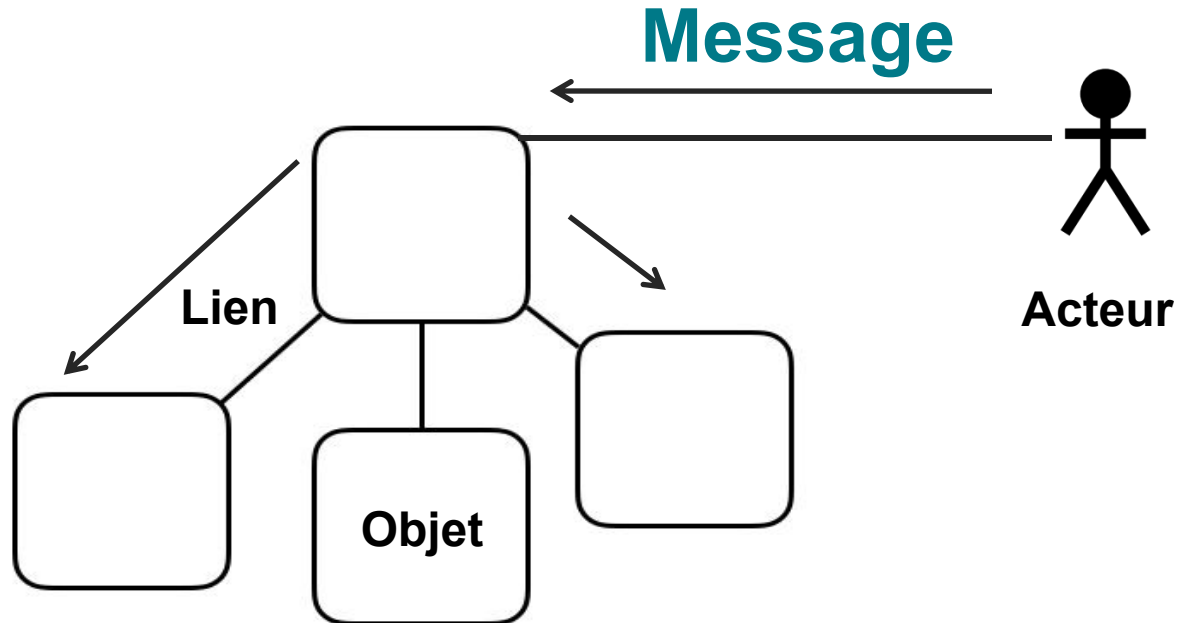
## Objet

- ❑ Le formalisme d'un lien dans le diagramme de collaboration est comme suivant:



# Diagramme de collaboration

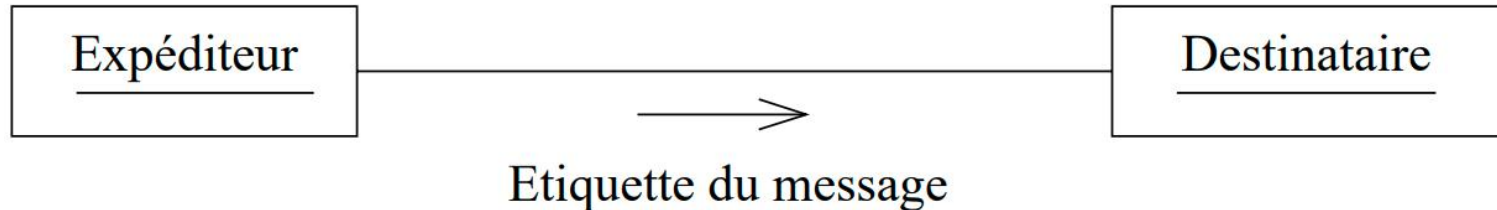
Le diagramme de collaboration comporte 4 concepts :



# Diagramme de collaboration

## Message

- ❑ Les objets communiquent en échangeant des messages représentés sous forme de flèches.
- ❑ Les messages sont étiquetés par le nom de l'opération ou du signal invoqué.
- ❑ L'envoi d'un message nécessite que le récepteur puisse réaliser une opération.



# Diagramme de collaboration

## Etiquette du Message

- ❑ Les étiquettes décrivent les messages auxquels elles sont attachées.
- ❑ La description d'une étiquette est faite par :
  - ❑ un nom
  - ❑ numéro d'une séquence (permet de préciser l'ordre d'émission des messages)
  - ❑ des arguments
  - ❑ un résultat attendu
  - ❑ une synchronisation
  - ❑ une condition d'émission.

# Diagramme de collaboration

Etiquette du Message: **Nom**

- ❑ C'est le nom de l'opération ou du signal invoqué par l'intermediaire de ce signal.

**Nom étiquette = Nom opération**

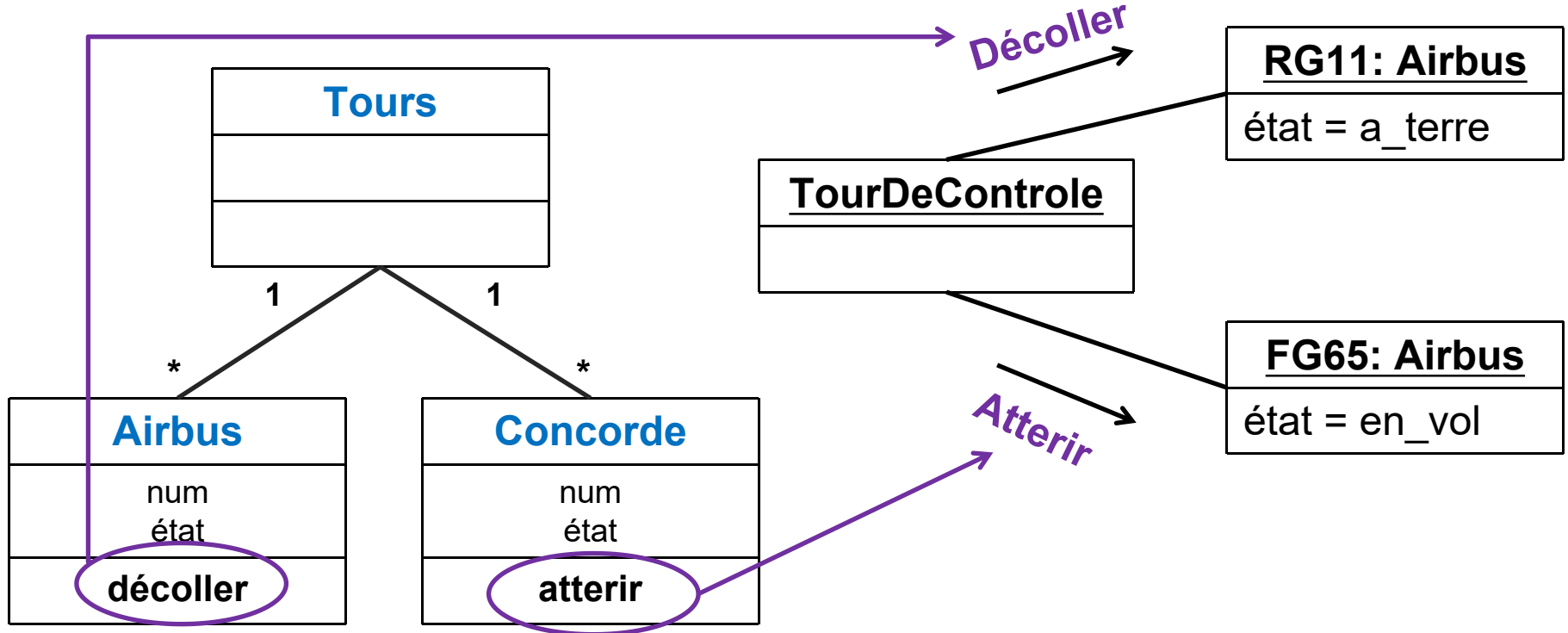
# Diagramme de collaboration

## Etiquette du Message: **Nom**

- ❑ Il doit exister une cohérence entre le diagramme de collaboration et le diagramme de classe :
  - ❑ Les opérations (noms des étiquettes) doivent être des méthodes dans les classes des objets récepteurs du messages.

# Diagramme de collaboration

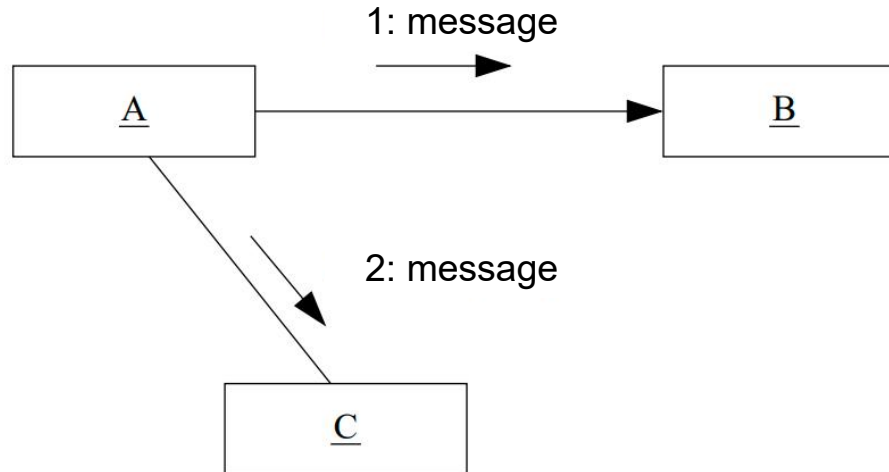
Etiquette du Message: **Nom**



# Diagramme de collaboration

## Etiquette du Message: N° Séquence

- ❑ Lorsque le diagramme de collaboration comporte plusieurs messages, il devient utile de représenter l'ordre d'envoi de ces messages.
- ❑ Ensemble de numéros ordonnant l'envoi des messages ( 1 puis 2 puis 3 ...)





# Diagramme de collaboration

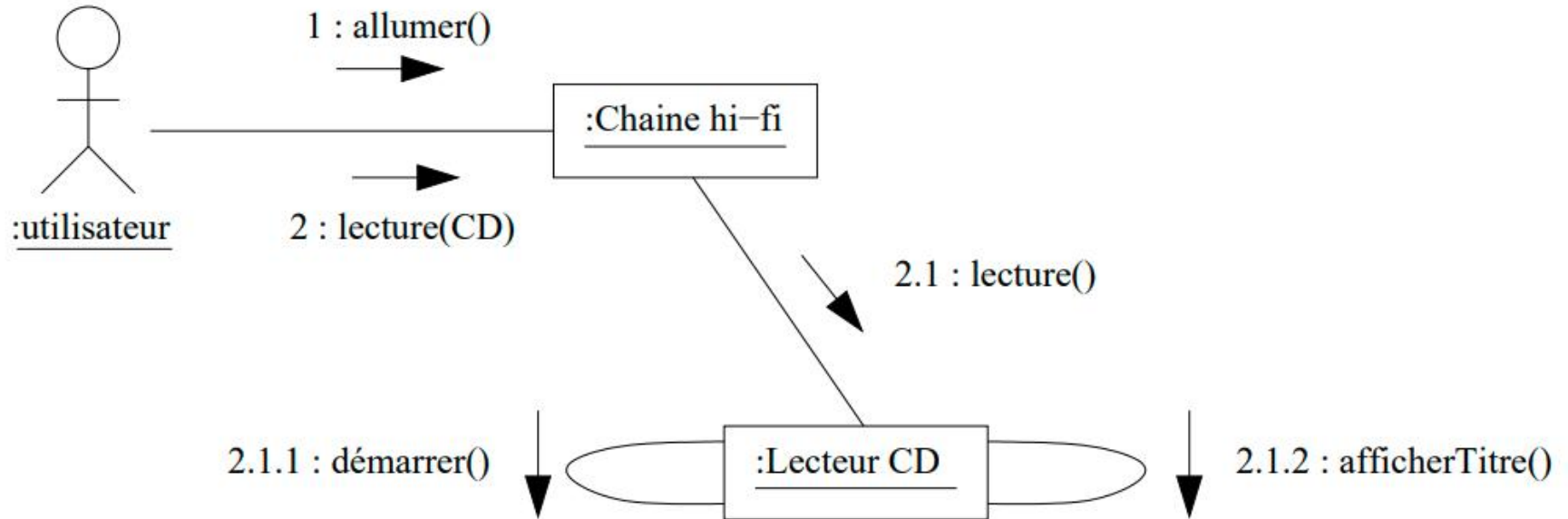
## Etiquette du Message: N° Séquence

- ❑ Dans un diagramme de collaboration, tous les messages ne correspondent pas à une seule activité. Il devient utile d'introduire:
  - ❑ Un partitionnement des messages (groupe de message)
  - ❑ Une hiérarchie de messages.
    - ❑ 1 (appel initial) puis 1.1 (premier appel imbriqué) puis 1.2 (second sous-appel) puis 2 (appel du même niveau que le numéro 1).

# Diagramme de collaboration

## Etiquette du Message: N° Séquence

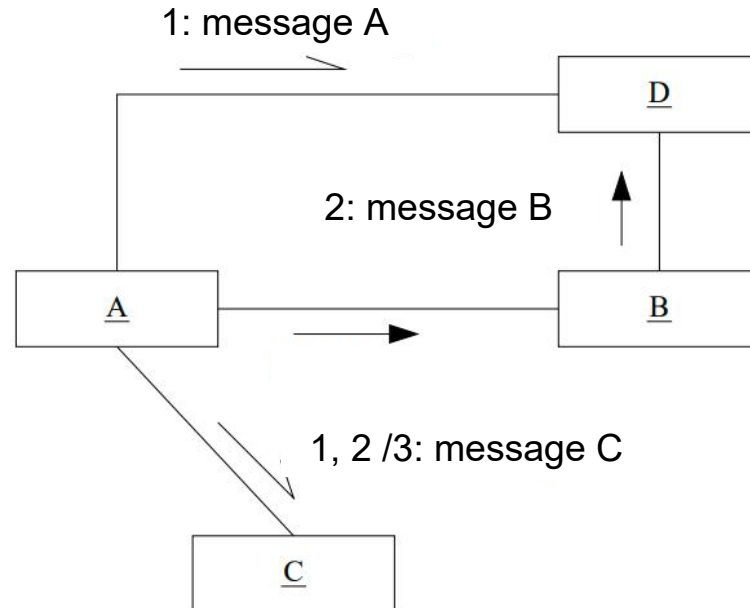
❑ Exemple:



# Diagramme de collaboration

## Etiquette du Message: **Synchronisation**

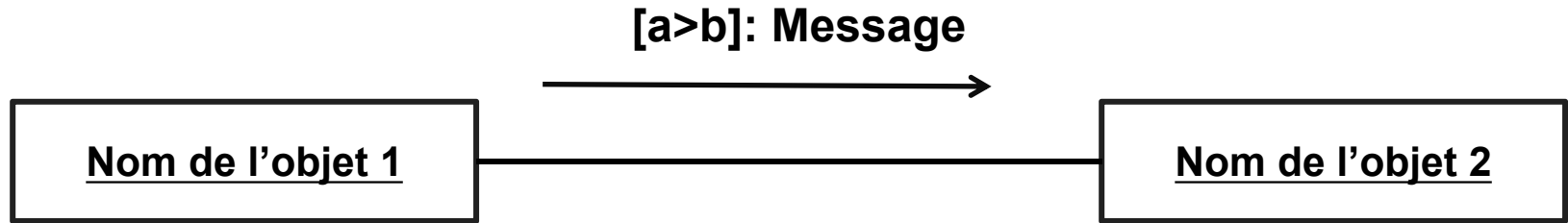
- ❑ Synchronization de message : envoi d'un message ssi d'autres messages ont **déjà** été envoyés.
- ❑ Syntaxe: **N° Séquence, N° Séquence ... /**



# Diagramme de collaboration

## Etiquette du Message: **Condition**

- ❑ L'envoi d'un message peut être soumis à une condition.
- ❑ La forme de la condition est: Condition : : = [ Condition]
- ❑ Exemple:

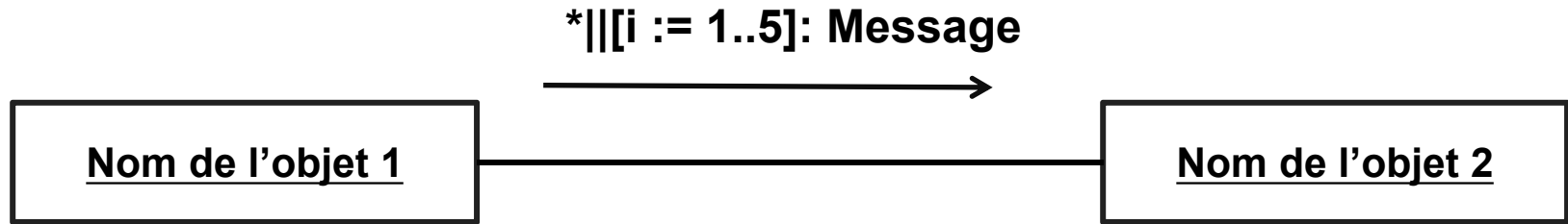


Ce message n'est envoyé que si  $a > b$

# Diagramme de collaboration

## Etiquette du Message: **Itération**

- ❑ L'envoi d'un message peut être répétitif.
- ❑ La forme de l'itération est: **\*|| [ Condition]**
- ❑ Exemple:

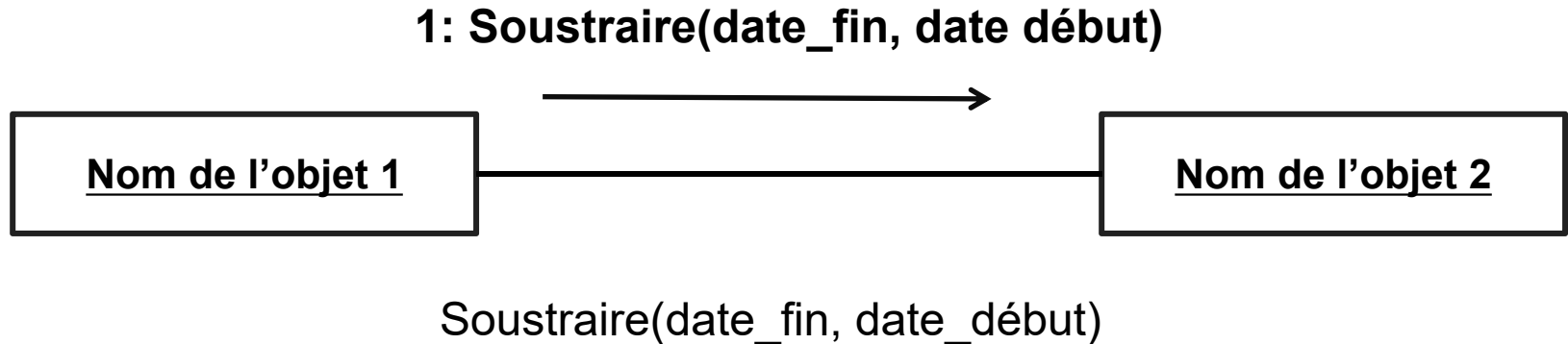


Représente l'envoi en parallèle de 5 messages

# Diagramme de collaboration

## Etiquette du Message: **Argument**

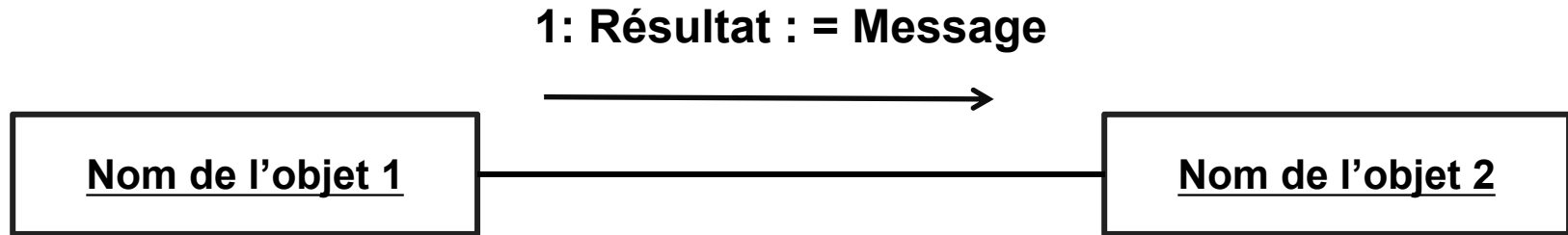
- ❑ L'appel d'un message peut préciser les arguments d'appel.
- ❑ La forme est la suivante: **Nom\_Message(arg1, arg2, ...)**
- ❑ Exemple:



# Diagramme de collaboration

## Etiquette du Message: **Résultat**

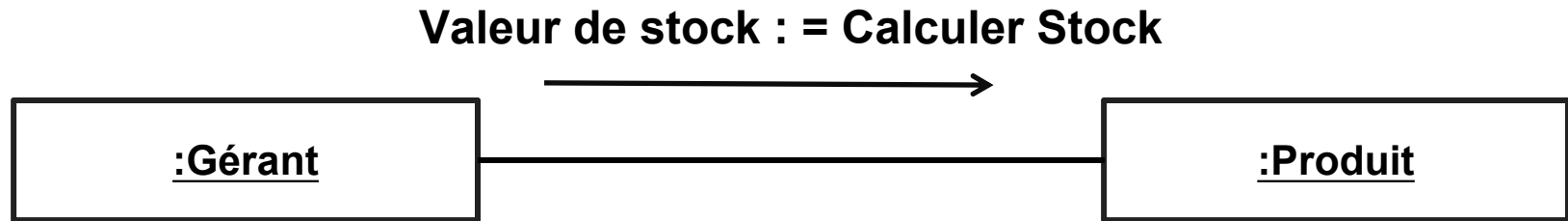
- ❑ Le resultat est constitué d'une liste de valeurs retournées par le message.
- ❑ Ces valeurs peuvent etre utilisées comme paramètres des autres messages.
- ❑ Formalisme:



# Diagramme de collaboration

Etiquette du Message: **Résultat**

❑ Exemple:





# Diagramme de collaboration

## Etiquette du Message: Règles Générales

- ❑ De façon générale la syntaxe d'une étiquette d'un message est la suivante :

```
[<Synchronisation '/'> [ '['<Condition>']' [<N° Séquence>] [ *[] [ '['<Itération>']' ] ] :]  
[<Résultat> :=] < Nom du Message >([<Arguments>])
```

# Diagramme de collaboration

## Etiquette du Message: Règles Générales

- ❑ De façon générale la syntaxe d'une étiquette d'un message est la suivante :

```
[Synchronisation '/' ] [ '['<Condition>']' [<N° Séquence>] [ *[] [] '['<Itération>']' ] :]  
[<Résultat> :=] <Nom du Message>([<Arguments>])
```

- ❑ Exemple 1:

**3 : bonjour()**

Ce message a pour numéro de séquence "3".

# Diagramme de collaboration

## Etiquette du Message: Règles Générales

- ❑ De façon générale la syntaxe d'une étiquette d'un message est la suivante :

```
[Synchronisation '/' ] [ '['<Condition>']' [<N° Séquence>] [ *[] [] '['<Itération>']' ] :]  
[<Résultat> :=] <Nom du Message>([<Arguments>])
```

- ❑ Exemple 2:

**[heure = midi] 1 : manger()**

Ce message n'est envoyé que s'il est midi.

# Diagramme de collaboration

## Etiquette du Message: Règles Générales

- ❑ De façon générale la syntaxe d'une étiquette d'un message est la suivante :

```
[Synchronisation '/' ] [ '['<Condition>']' [<N° Séquence>] [ *[[ ] ['<Itération>']] ] :]  
[<Résultat> :=] <Nom du Message>([<Arguments>])
```

- ❑ Exemple 3:

**3 / \*[[i := 1..9] : fermer()**

Représente l'envoi en parallèle de 9 messages. Ces messages ne seront envoyés qu'après l'envoi du message 3.

# Diagramme de collaboration

## Etiquette du Message: Règles Générales

- ❑ De façon générale la syntaxe d'une étiquette d'un message est la suivante :

```
[Synchronisation '/' ] [ '['<Condition>']' [<N° Séquence>] [ *[] [] '['<Itération>']' ] :]  
[<Résultat> :=] <Nom du Message>([<Arguments>])
```

- ❑ Exemple 4:

**1.3 \* : ouvrir()**

Ce message est envoyé de manière séquentielle un certain nombre de fois.

# Diagramme de collaboration

## Etiquette du Message: Règles Générales

- ❑ De façon générale la syntaxe d'une étiquette d'un message est la suivante :

```
[Synchronisation '/' ] [ '['<Condition>']' [<N° Séquence>] [ *[] [] '['<Itération>']' ] :]  
[<Résultat> :=] <Nom du Message>([<Arguments>])
```

- ❑ Exemple 5:

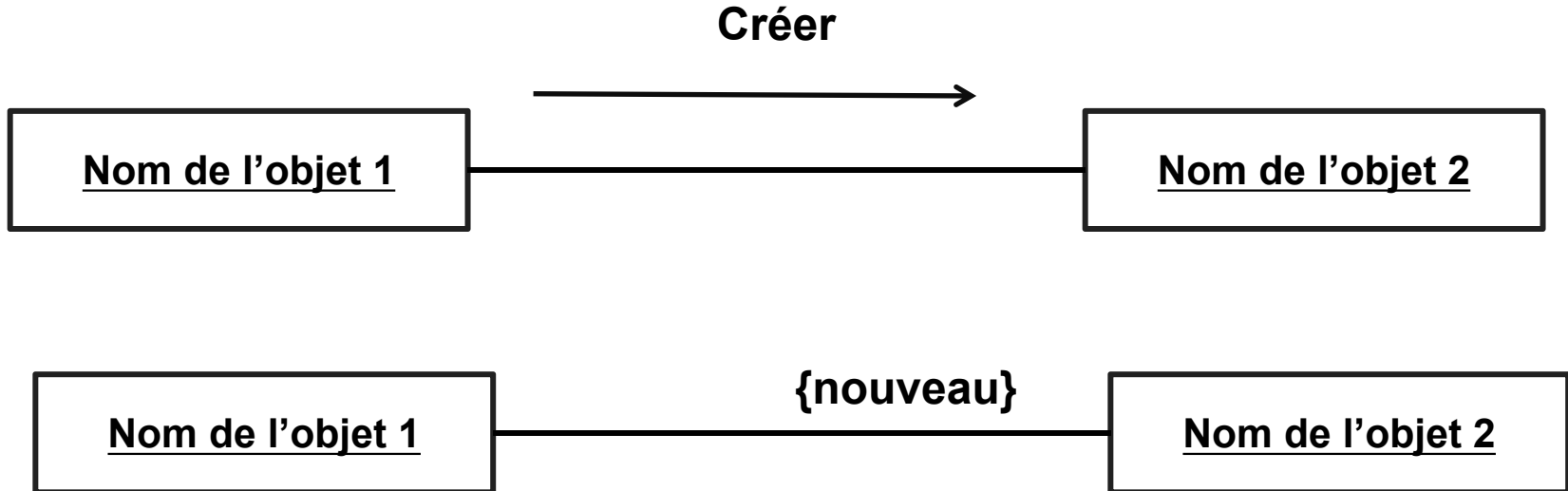
**1.3,2.1 / [t < 10s] 2.5 : age := demanderAge(nom,prenom)**

Ce message (numéro 2.5) ne sera envoyé qu'après les messages 1.3 et 2.1, et que si "t < 10s".

# Diagramme de collaboration

## Création et destruction d'objets

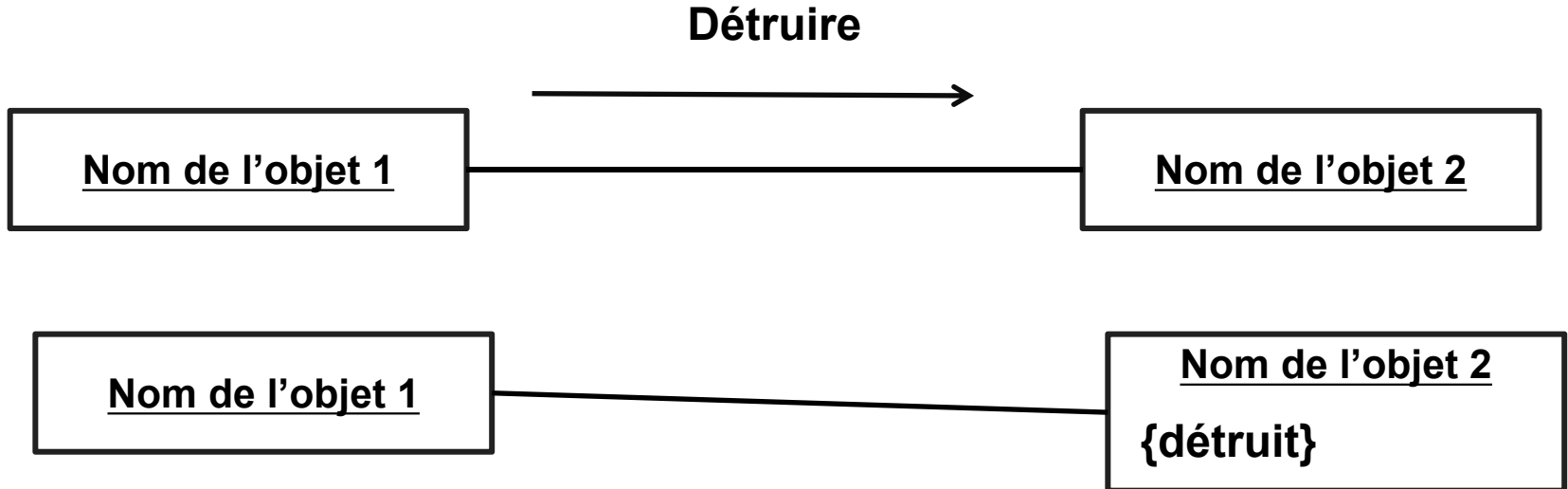
- ❑ Un message peut créer ou détruire un autre objet.
- ❑ Création d'objet:



# Diagramme de collaboration

## Création et destruction d'objets

- ❑ Un message peut créer ou détruire un autre objet.
- ❑ Destruction d'objet:





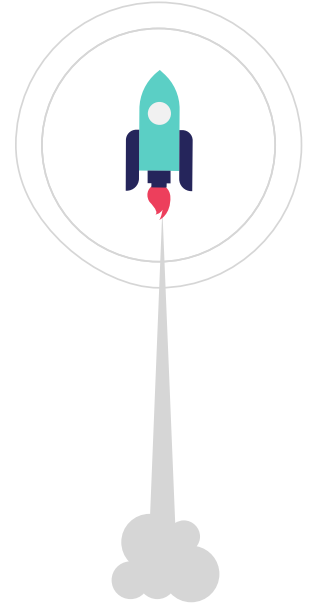
# Etude de cas

On souhaite gérer les différents objets qui concourent à l'activité d'un magasin de vente de fleurs.

- ☐ Le client demande au vendeur des renseignements sur les compositions florales.
- ☐ Le vendeur lui fournit toutes les informations nécessaires.
- ☐ Le client commande alors la composition de son choix et le vendeur émet le bon de fabrication qu'il transmet à son ouvrier fleuriste.
- ☐ Le vendeur édite ensuite la facture correspondante.
- ☐ L'ouvrier fleuriste crée la composition puis archive le bon de fabrication.
- ☐ Il remet alors la composition au vendeur.
- ☐ La facture est remise au client pour règlement une fois le bouquet réalisé.
- ☐ Une fois la facture réglée, le client récupère sa composition et quitte le magasin.

# **Vision Dynamique du système :**

## **Diagramme de Séquence**



# Diagramme de Séquence

## Introduction

- ❑ Le diagramme de séquence est une variante du diagramme de collaboration, même type de diagramme (**vue dynamique** + **interaction d'objets pour réaliser un scénario**).
- ❑ Il se base sur l'aspect temporel.

# Diagramme de Séquence

## Introduction

(Vue dynamique + Interaction d'objets pour réaliser un scénario)

dans un aspect temporel

=

Décrire la réalisation des cas d'utilisation sur le système

décrit par le diagramme de classes

# Diagramme de Séquence

## Introduction

- ❑ Un diagramme de cas d'utilisation répond à la question **QUOI ?**
  - ❑ Qui ce qu'on va réaliser dans le système ? (les interactions possibles entre le système et les acteurs).
- ❑ Un diagrammes de classes répond à la question **Qui?**
  - ❑ QUI sera à l'oeuvre dans le système pour réaliser les fonctionnalités décrites par le diagramme de cas d'utilisation ? ( la structure et les liens entre les objets d'un système).

Le mélange entre **QUOI ?** et **QUI ?** va répondre à la question **Comment ?**

# Diagramme de Séquence

## Introduction

- ❑ Le diagramme de séquence répond à la question **COMMENT ?**
  - ❑ Comment les **objets** du système interagissent entre eux et avec les **acteurs**

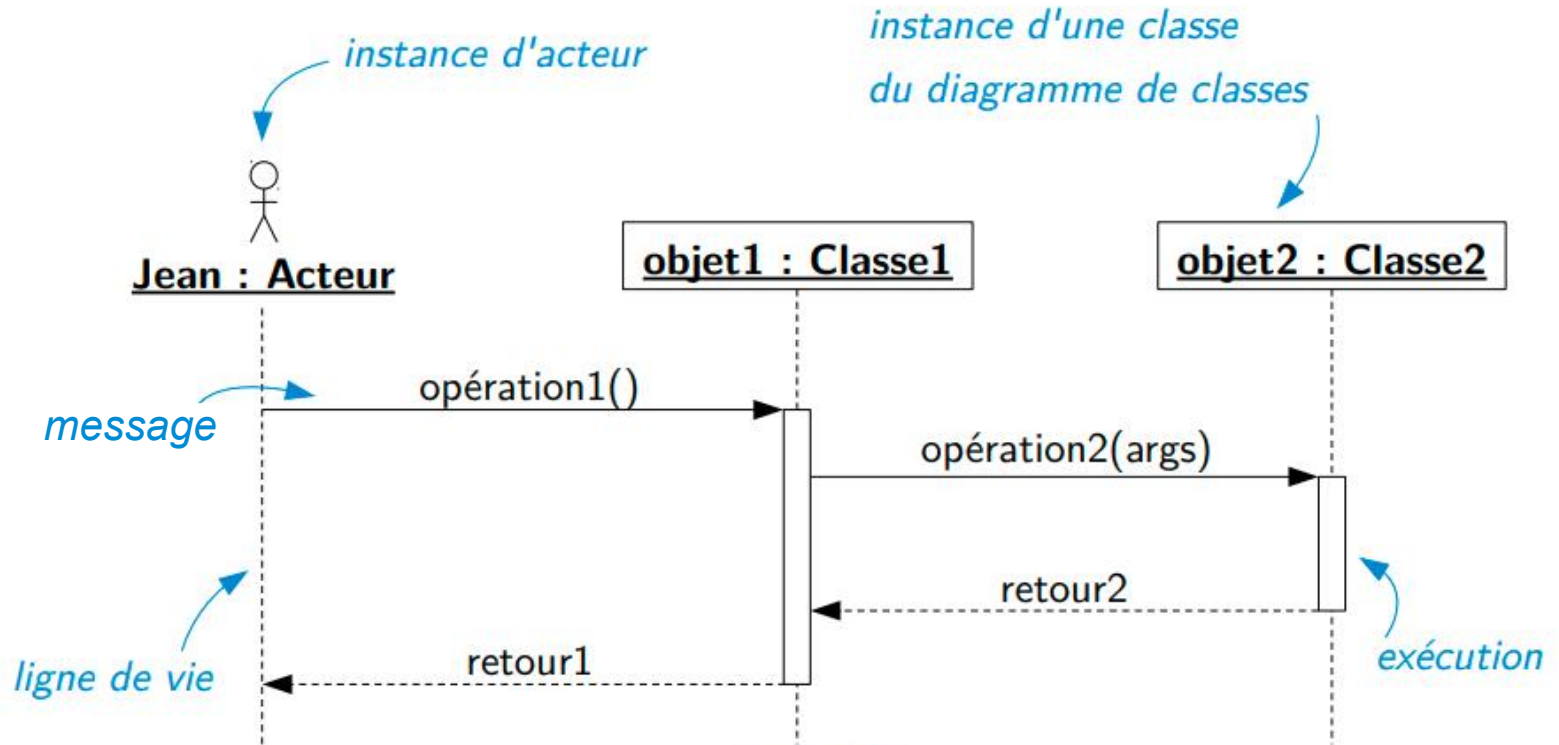
Toute personne ou tout objet qui interagit à l'intérieur du système

Toute personne ou tout objet qui interagit avec le système

Un objet ou un acteur peut s'agir d'êtres humains, des ordinateurs ou des systèmes logiciels.

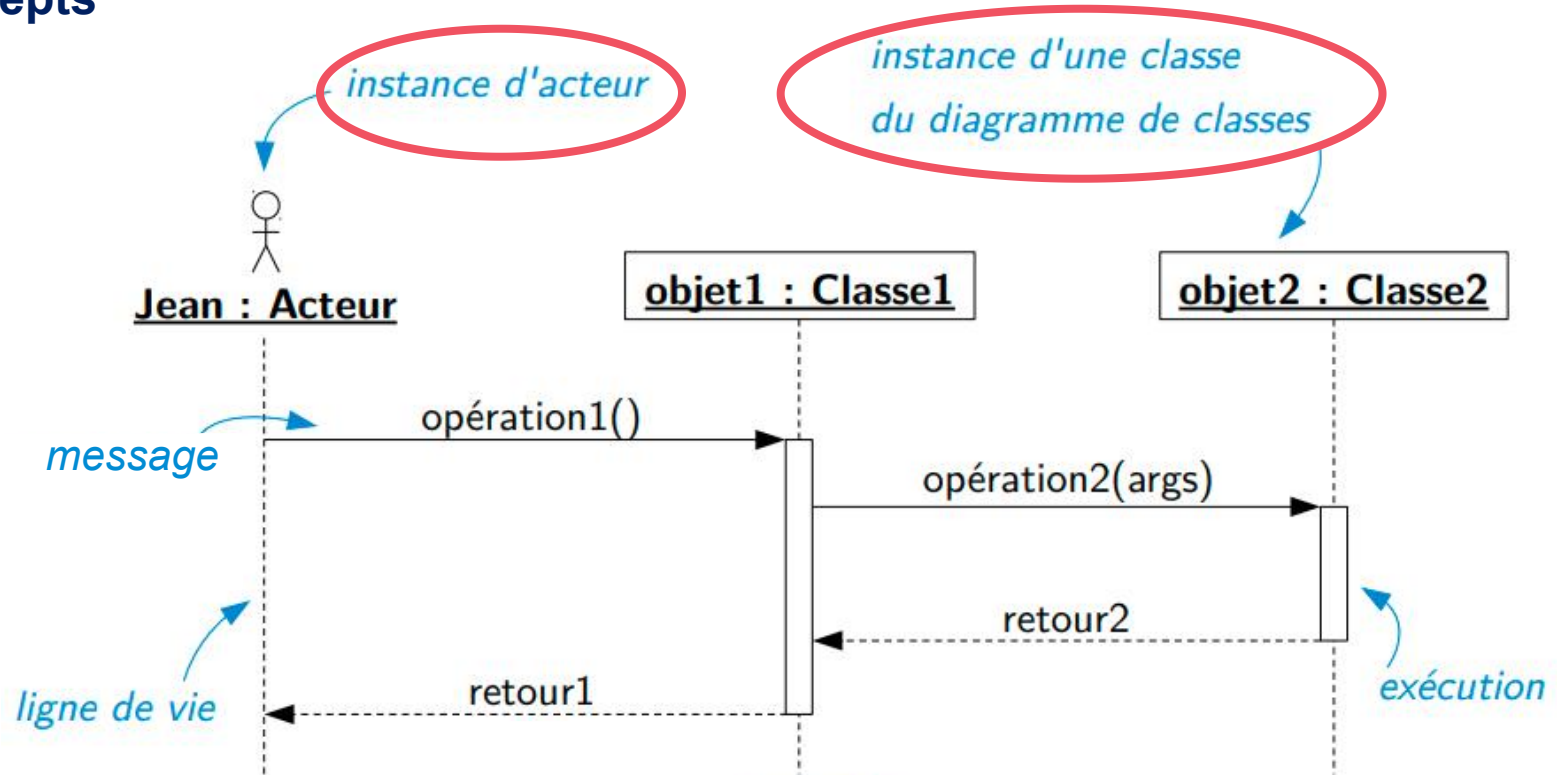
# Diagramme de Séquence

## Concepts



# Diagramme de Séquence

## Concepts

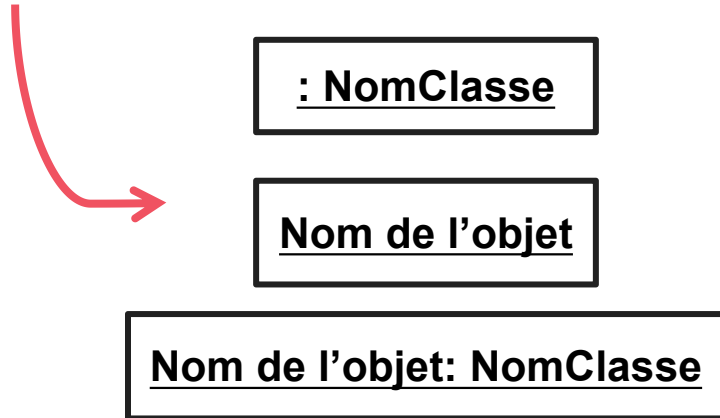




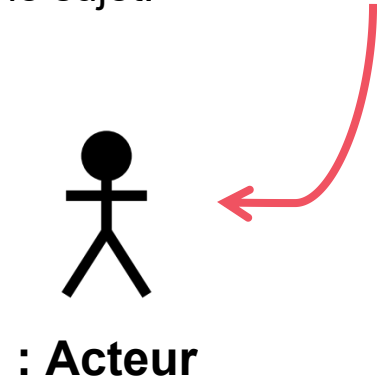
# Diagramme de Séquence

## Objet / Acteur

- ❑ **Les objets** sont des éléments de modèle qui représentent des instances d'une ou plusieurs classes.



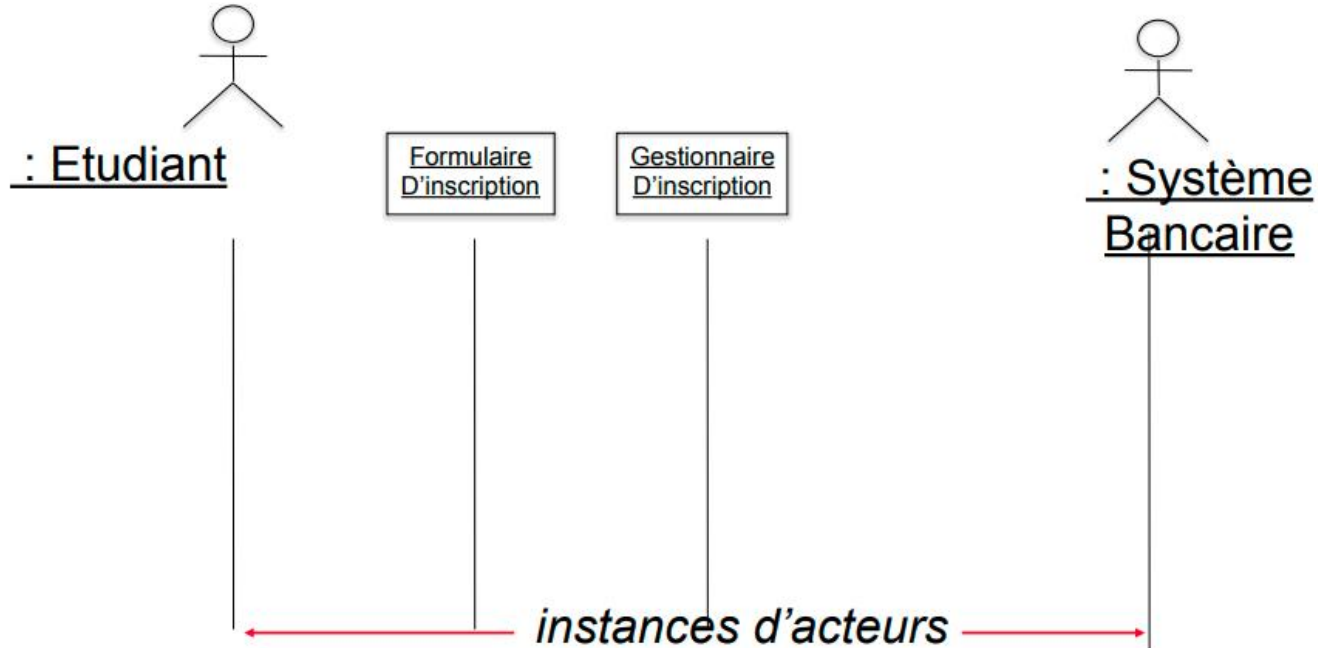
- ❑ **L'acteur** spécifie un rôle joué par un utilisateur ou tout autre système qui interagit avec le sujet.



# Diagramme de Séquence

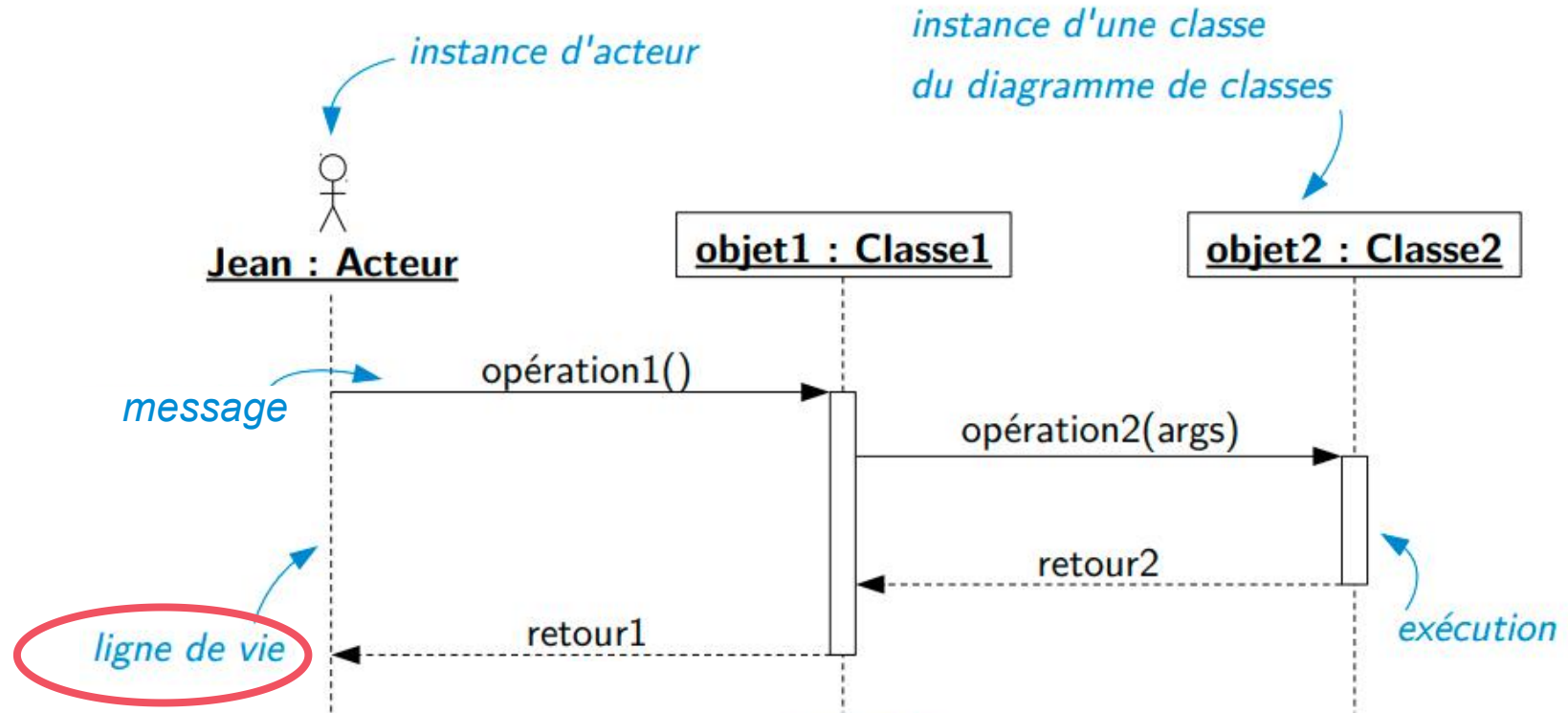
## Objet / Acteur

### Exemple:



# Diagramme de Séquence

## Concepts



# Diagramme de Séquence

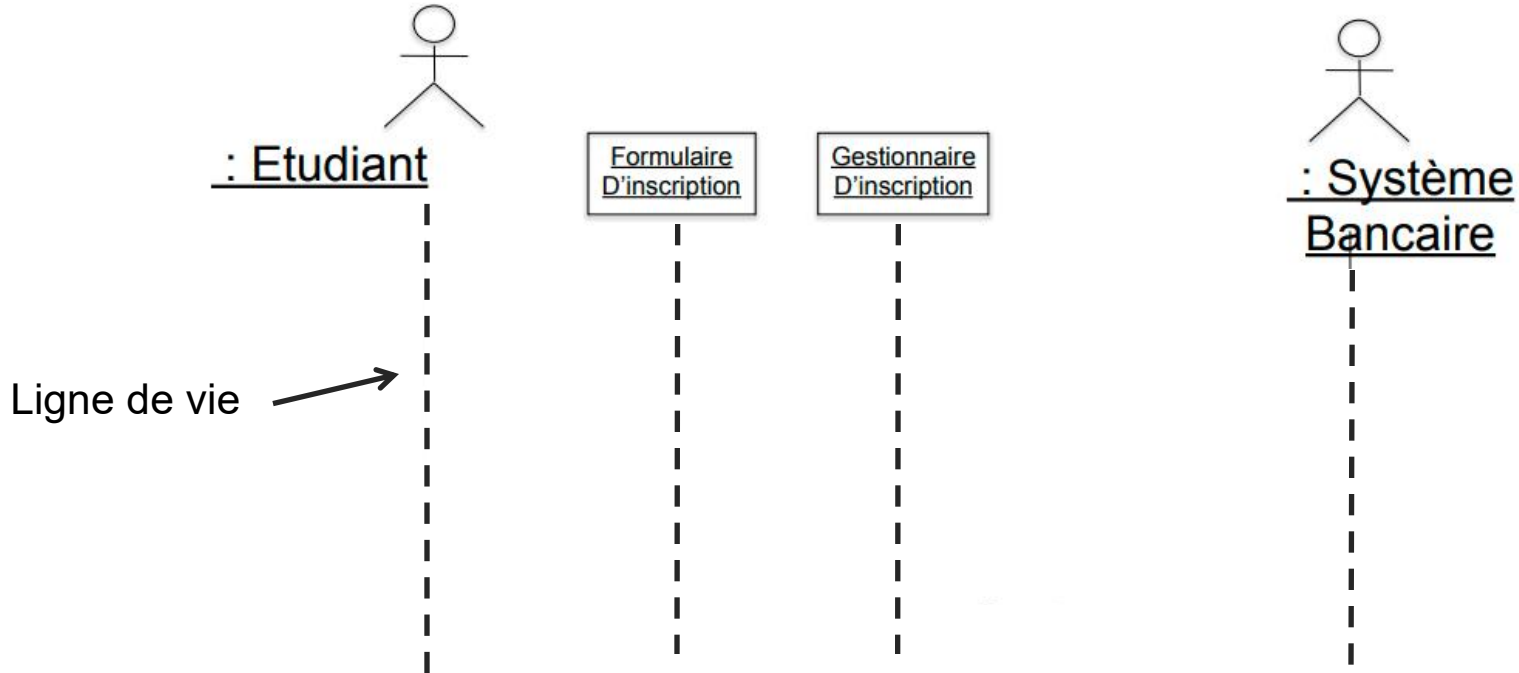
## Ligne de vie

- ❑ La ligne de vie représente le temps ainsi que la durée de vie des objets.
- ❑ La ligne de vie des objets est représentée par une ligne verticale en traits pointillés, placée sous le symbole de l'objet concerné. Cette ligne de vie précise l'existence de l'objet concerné durant un certain laps de temps.

# Diagramme de Séquence

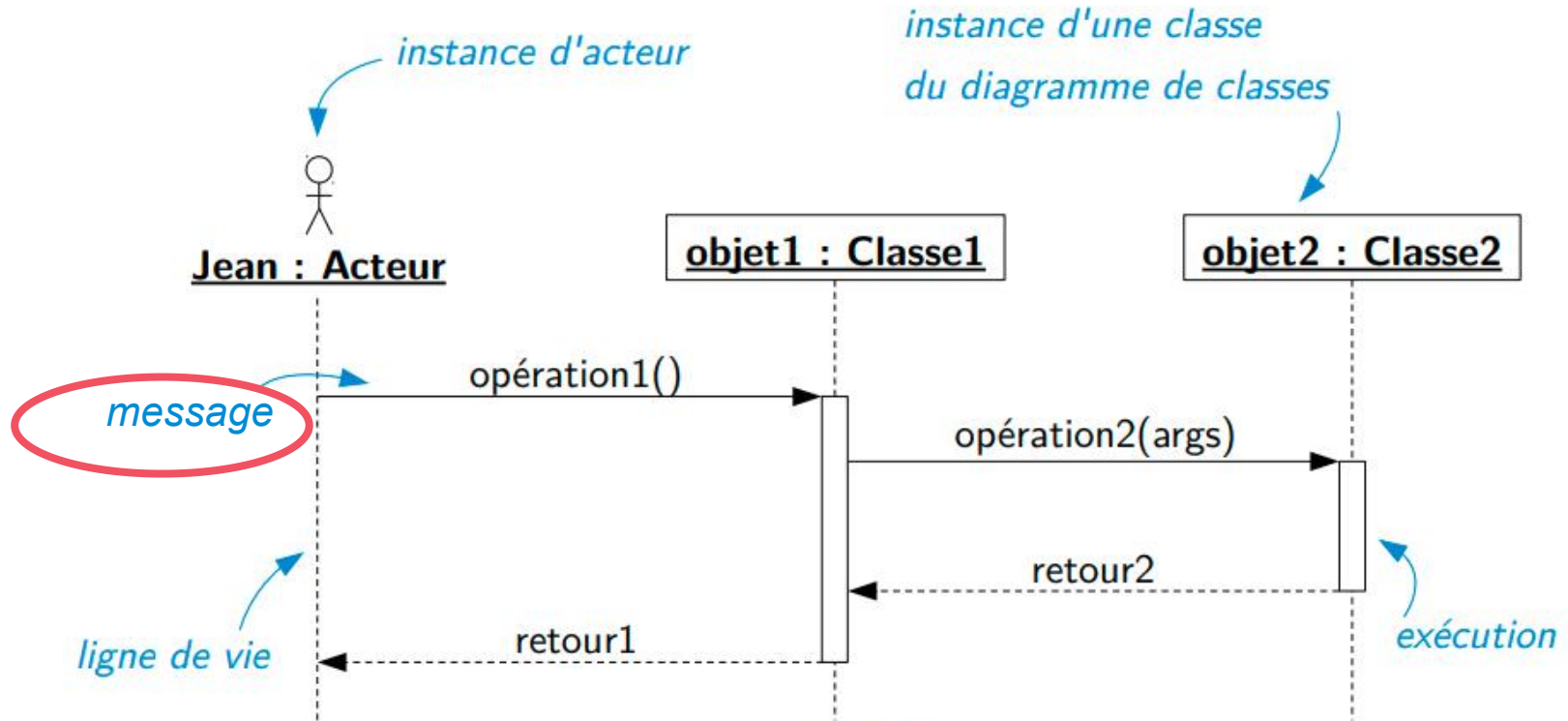
## Ligne de vie

### Exemple:



# Diagramme de Séquence

## Concepts



# Diagramme de Séquence

## Message

- ❑ Les messages sont représentés par des flèches directionnelles.
- ❑ Cette représentation est similaire à celle d'une association sur un diagramme de classes ; cependant, les messages servent à représenter la communication entre les objets, et non la relation structurelle présente entre les classes.
- ❑ Au-dessus des flèches directionnelles figure un texte informant du message envoyé entre les objets.

### Exemple:



# Diagramme de Séquence

## Message

- ❑ Les messages peuvent être numérotés (N° de séquence) selon l'ordre dans lequel ils sont envoyés. Sur les diagrammes de séquence, cette numérotation est quelque peu redondante, car l'ordre des messages est toujours déterminé par l'ordre dans lequel ils apparaissent dans le diagramme, du haut vers le bas.

### Exemple:



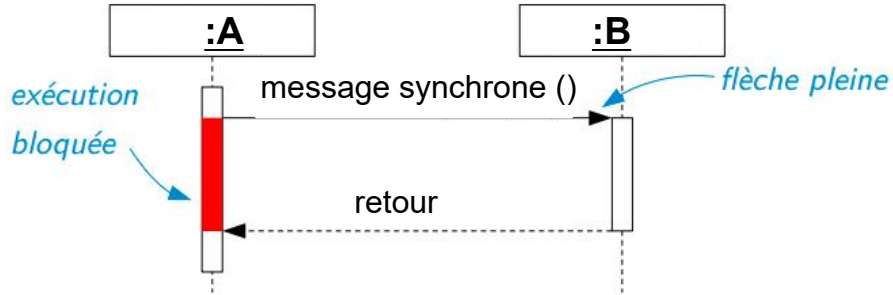


# Diagramme de Séquence

## Types de messages: (deux types)

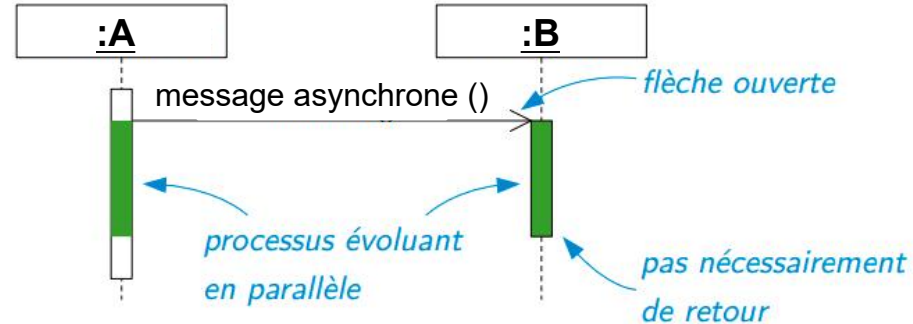
### ❑ Message synchrone :

Émetteur bloqué en attente du retour.



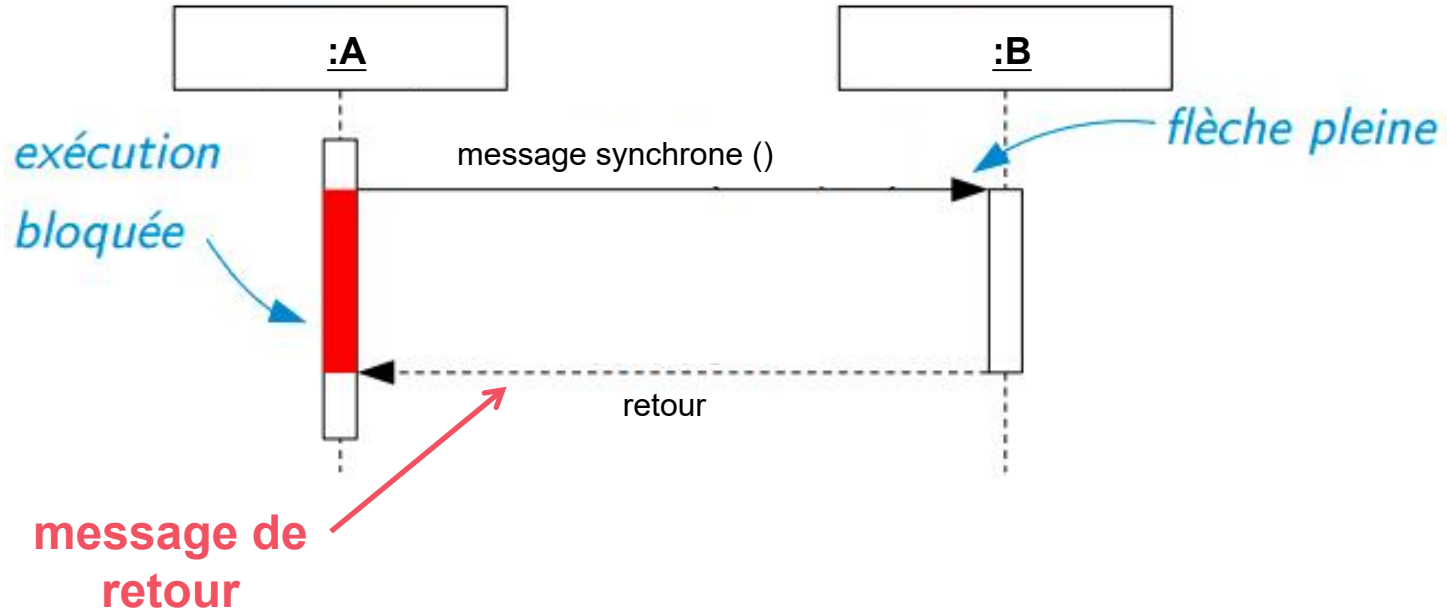
### ❑ Message asynchrone :

Émetteur non bloqué, continue son exécution.



# Diagramme de Séquence

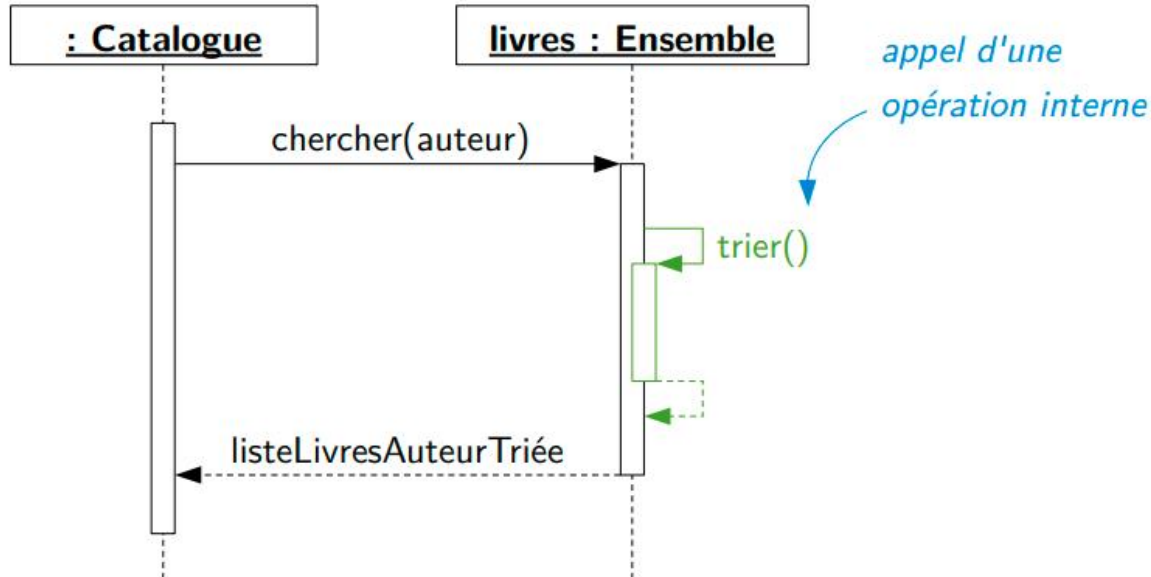
## Syntaxe de réponse à un message synchrone



# Diagramme de Séquence

Envoi de messages d'un objet sur lui même: (Message réflexif)

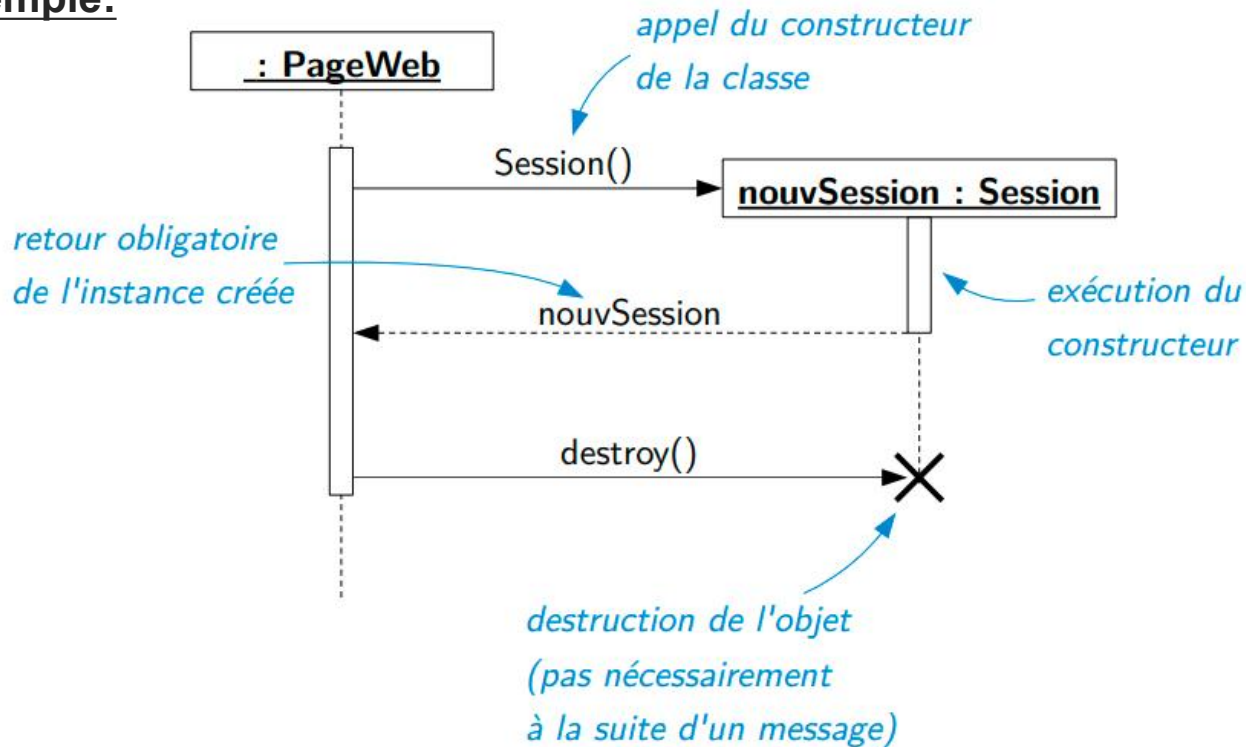
Explication avec exemple:



# Diagramme de Séquence

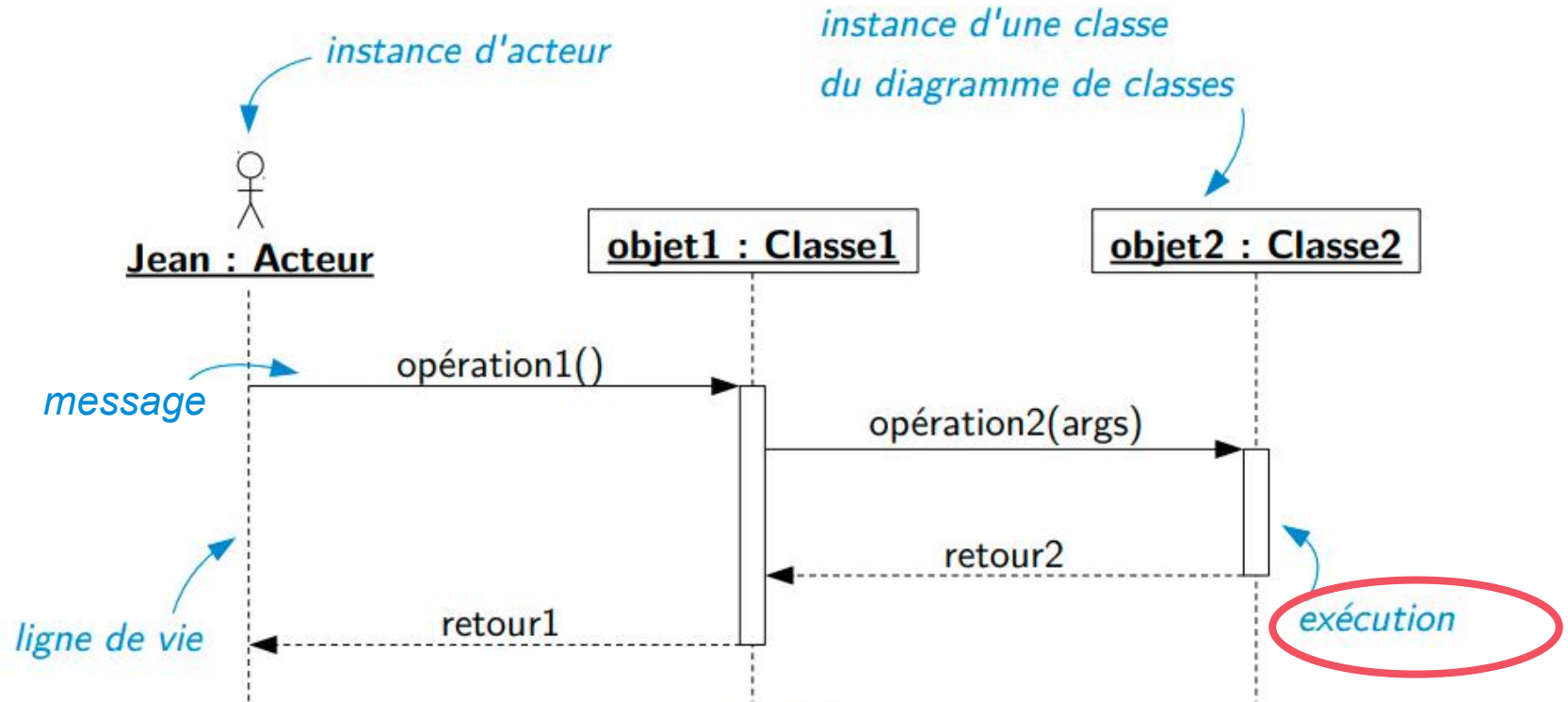
## Création et destruction d'un objet par message

### Explication avec exemple:



# Diagramme de Séquence

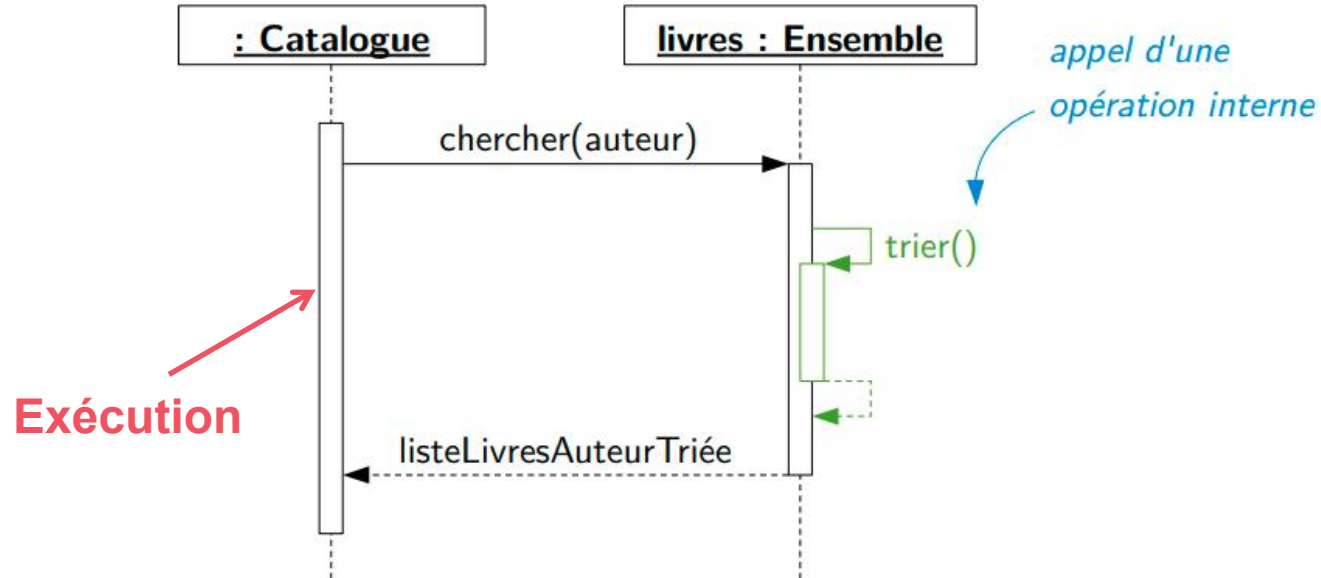
## Concepts



# Diagramme de Séquence

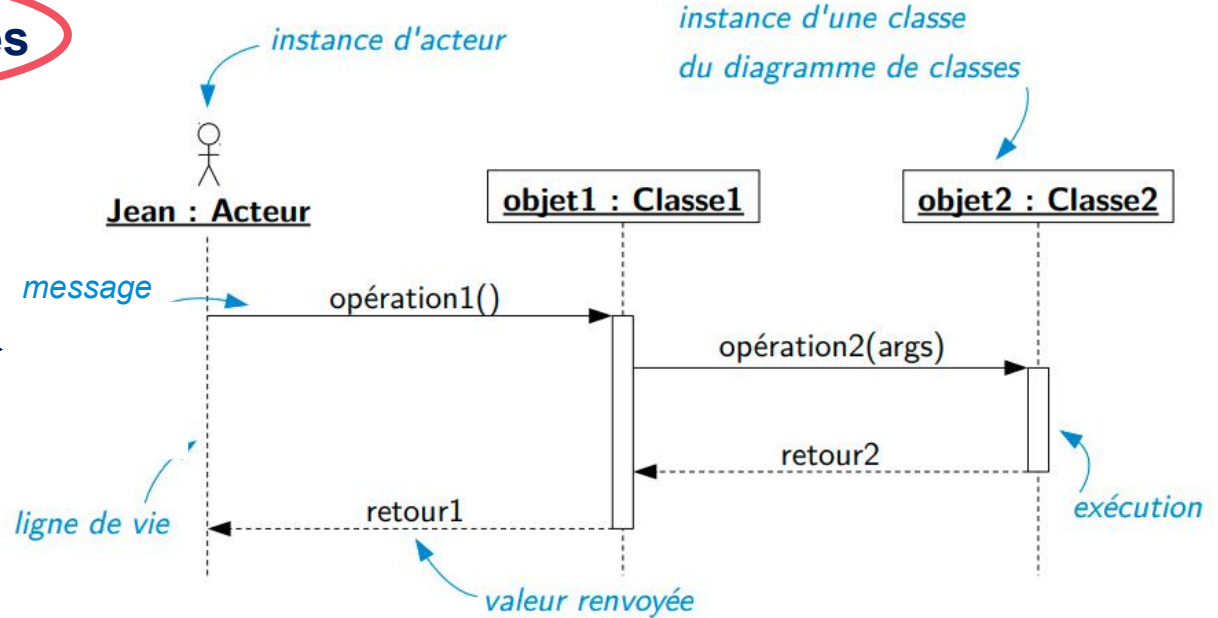
## Exécution

- ❑ C' est la période d'activité des objets. Elles présenter par un rectangle.



# Diagramme de Séquence

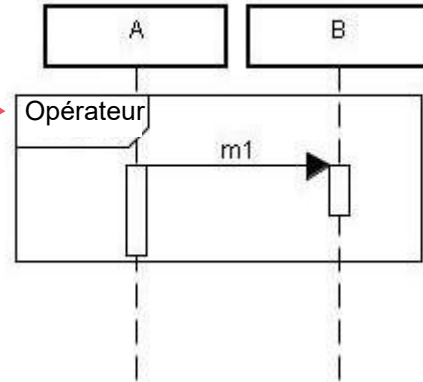
Les fragments combinés



# Diagramme de Séquence

## Les fragments combinés

- ❑ UML 2.0 a introduit les fragments combinés dans les diagrammes de séquence. Ils permettent de mieux décrire les scénarios.
- ❑ Un fragment combiné se représente de la même façon qu'une interaction. Il est représenté dans un rectangle dont le coin supérieur gauche contient le type de la combinaison, appelé opérateur d'interaction.





# Diagramme de Séquence

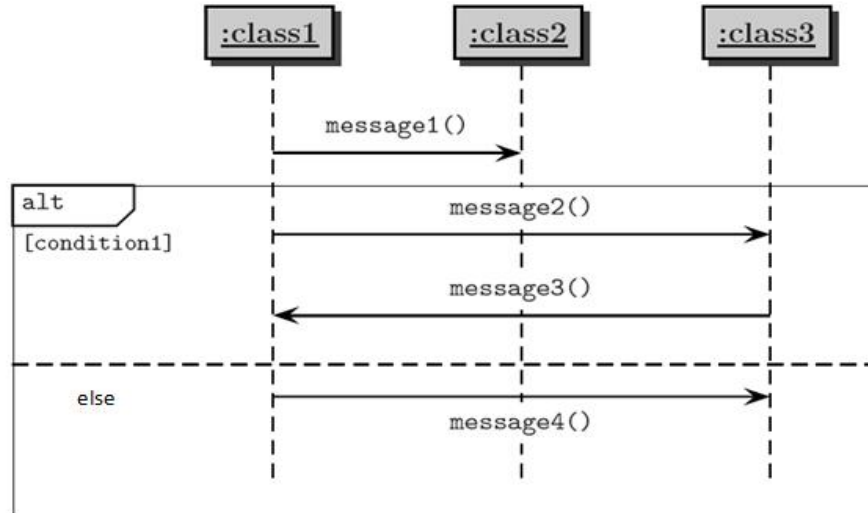
## Les fragments combinés

- ❑ Les opérateurs d'interaction:
  - ❑ Opérateur alternative **alt**
  - ❑ L'opérateur **opt**
  - ❑ L'opérateur **break**
  - ❑ L'opérateur **par**
  - ❑ L'opérateur **Loop**
  - ❑ L'opérateur **Référence**

# Diagramme de Séquence

## Les fragments combinés: Opérateur alternative "alt"

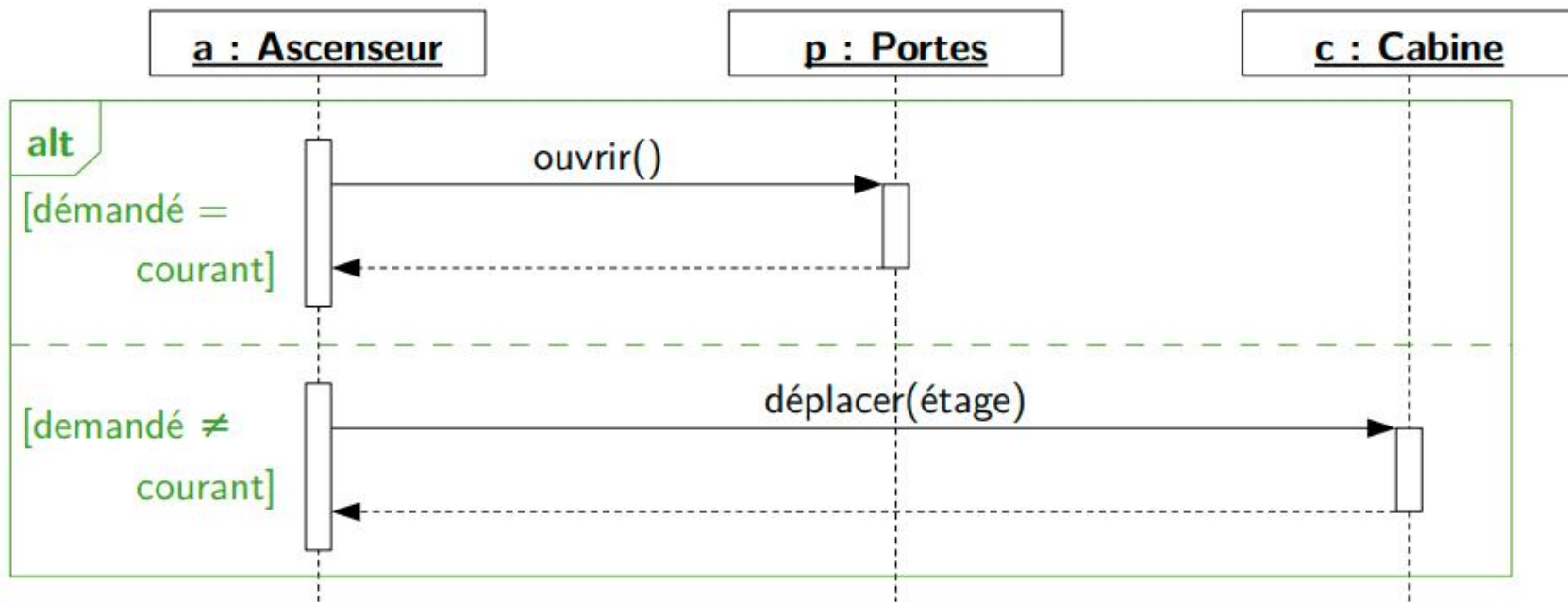
L'opérateur "alt" désigne un choix, une alternative. Il représente deux comportements possibles : c'est en quelque sorte l'équivalent du **SI...ALORS...SINON**



# Diagramme de Séquence

## Les fragments combinés: Opérateur alternative "alt"

### Exemple:



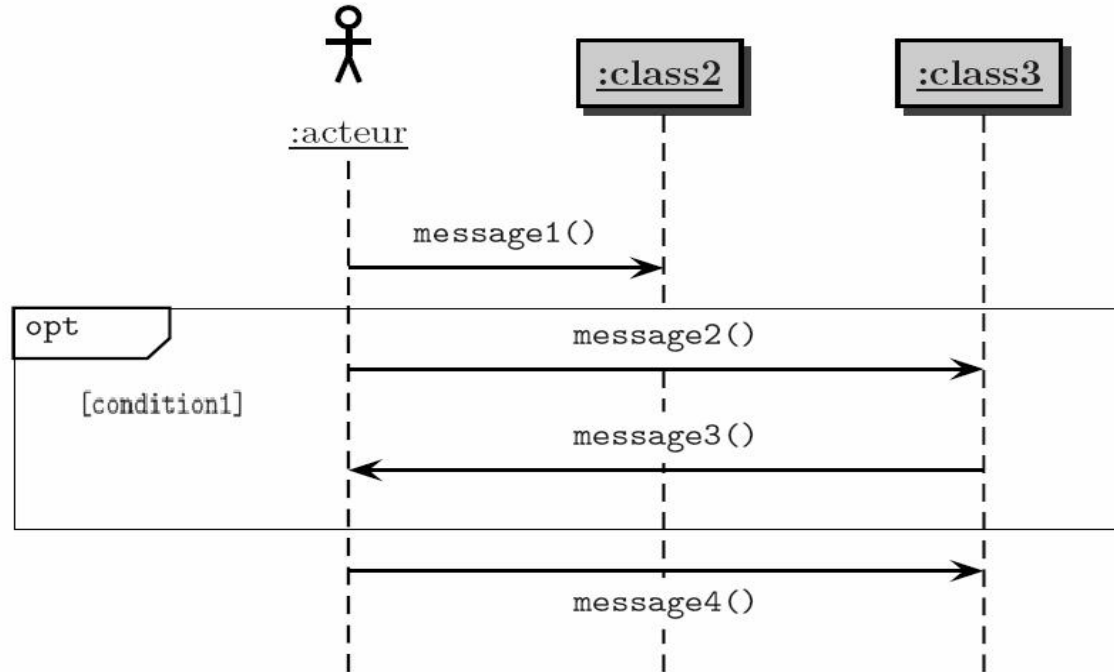
# Diagramme de Séquence

## Les fragments combinés: Opérateur "opt"

- ❑ L'opérateur "**opt**" désigne un fragment combiné optionnel comme son nom l'indique : c'est à dire qu'il représente un comportement qui peut se produire... ou pas.
- ❑ Un fragment optionnel est équivalent à un fragment "alt" qui ne posséderait pas la partie else (une seule branche). Un fragment optionnel est donc une sorte de **SI...ALORS**.

# Diagramme de Séquence

## Les fragments combinés: Opérateur "opt"



# Diagramme de Séquence

## Les fragments combinés: L'opérateur "break"

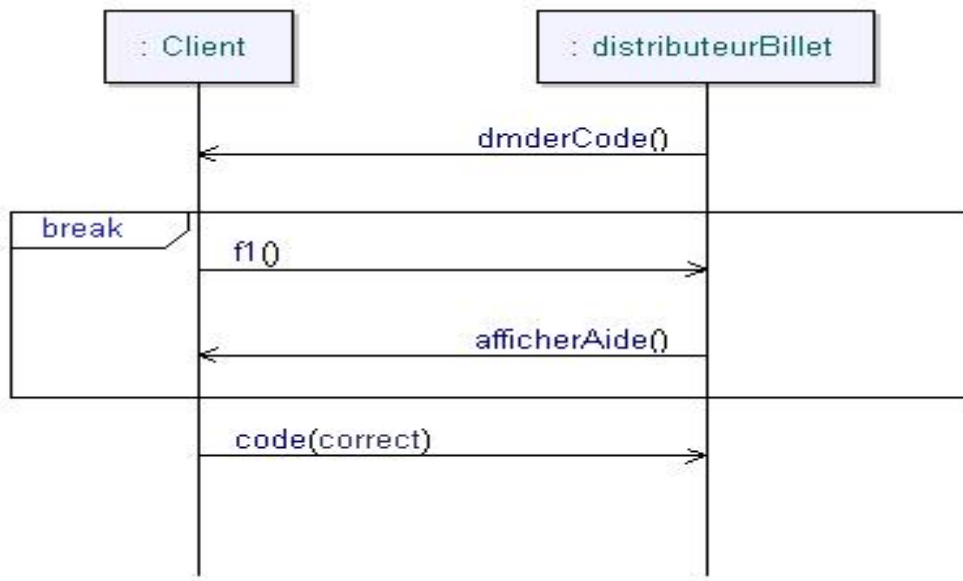
- ❑ L'opérateur "break" est utilisé dans les scénarii d'exception. Les interactions de ce fragment seront exécutées à la place des interactions décrites en dessous. Il y a donc une notion d'interruption du flot "normal" des interactions.

# Diagramme de Séquence

## Les fragments combinés: L'opérateur "break"

### Exemple:

- le distributeur demande à l'utilisateur son code.
- l'utilisateur peut choisir de rentrer son code ou de consulter l'aide.
- S'il choisit de consulter l'aide, l'interaction relatif à la saisie du code est interrompu. Les interactions de l'opérateur break sont "exécutées".



# Diagramme de Séquence

## Les fragments combinés: L'opérateur "par"

- ❑ L'opérateur "par" est utilisé pour représenter des interactions ayant lieu en parallèle.

### Exemple:

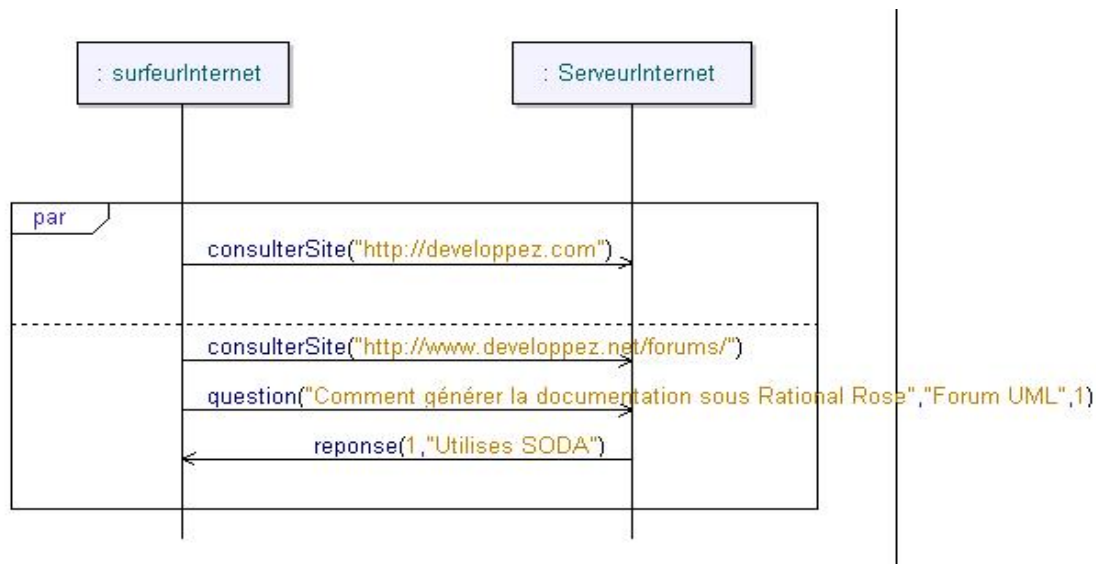
Un développeur ayant accès à Internet

peut consulter en parallèle le site

*<http://www.developpez.com>*

et le site

*<http://www.developpez.net/forums/>*





# Diagramme de Séquence

## Les fragments combinés: L'opérateur "Loop"

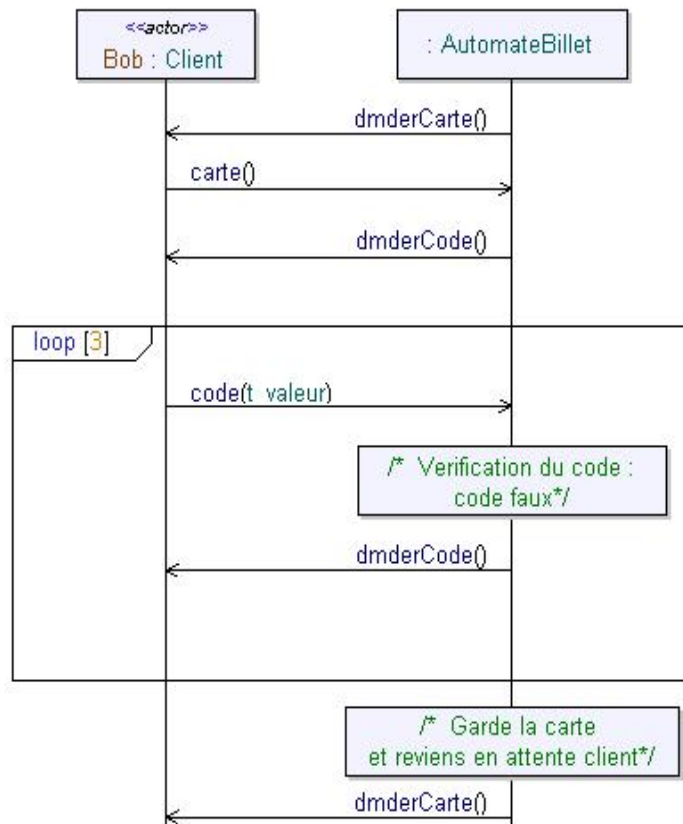
- ❑ L'opérateur "loop" (boucle) est utilisé pour décrire un ensemble d'interaction qui s'exécutent en boucle. En général, une contrainte appelée "garde" indique le nombre de répétitions (minimum et maximum) ou bien une condition booléenne à respecter:
  - ❑ exemple de garde : [3], [1,4], [t<10]

# Diagramme de Séquence

## Les fragments combinés: L'opérateur "Loop"

### Exemple:

Le diagramme de séquence indique que lorsque l'utilisateur se trompe trois fois de code, la carte est gardée et le distributeur se remet en mode d'attente d'une carte.



# Diagramme de Séquence

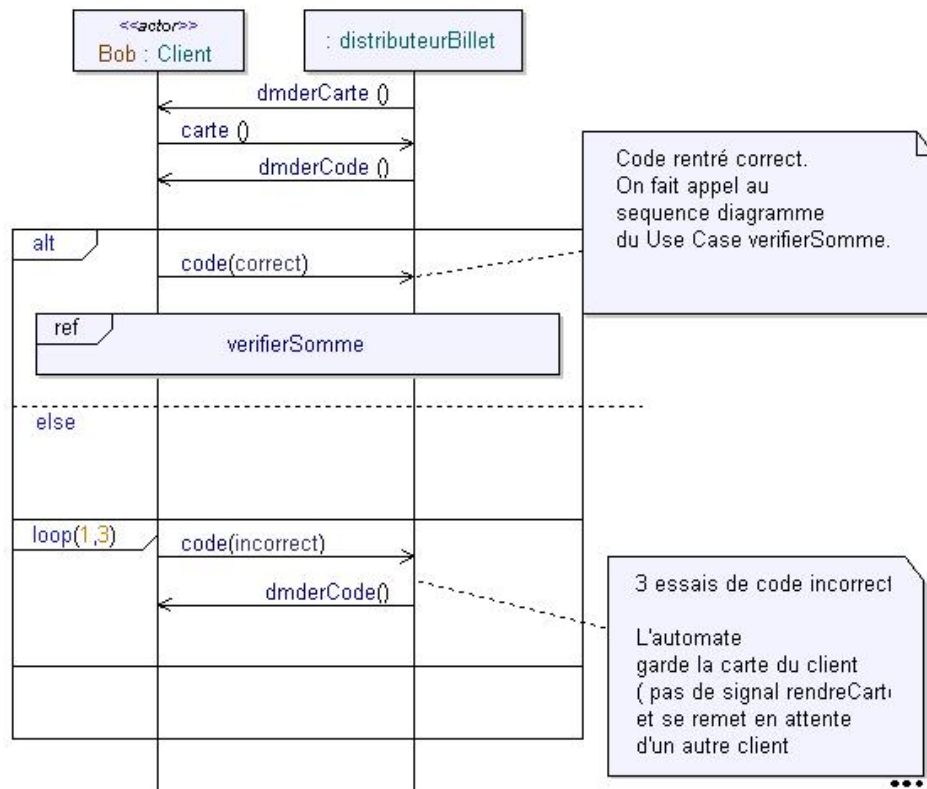
## Les fragments combinés: Référence

- ❑ Une référence peut être vue comme un pointeur ou un raccourci vers un autre diagramme de séquence existant. Cela équivaut à copier le contenu du diagramme de séquence pointé en lieu et place de la référence.

# Diagramme de Séquence

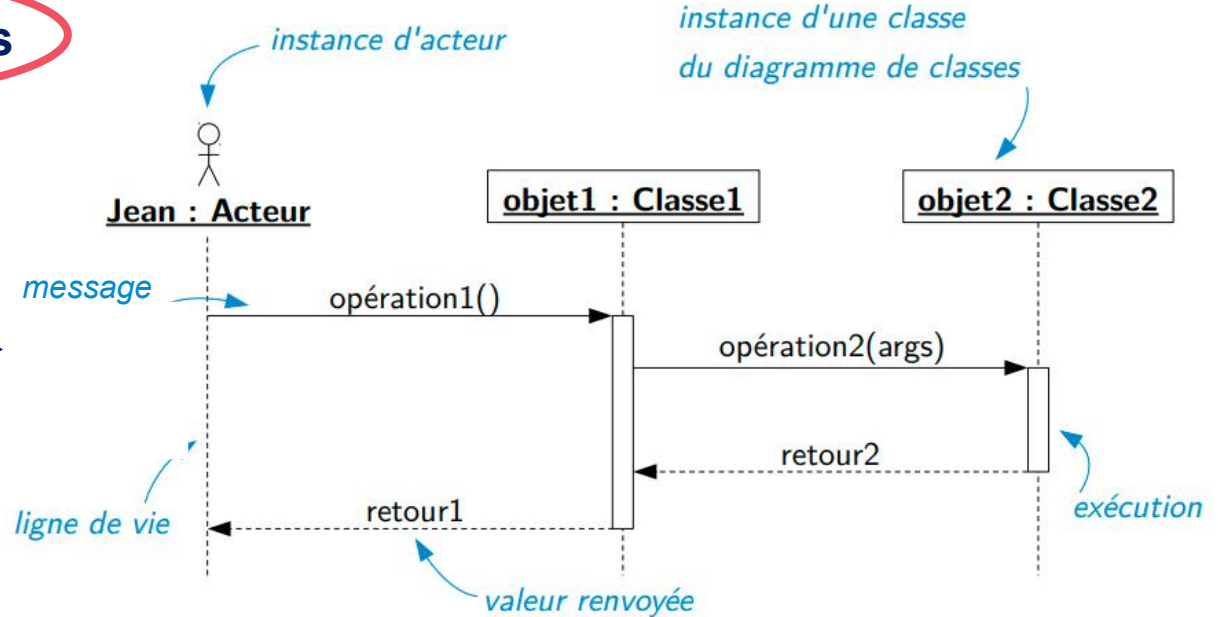
## Les fragments combinés: Référence

### Exemple:



# Diagramme de Séquence

## Contraintes temporelles



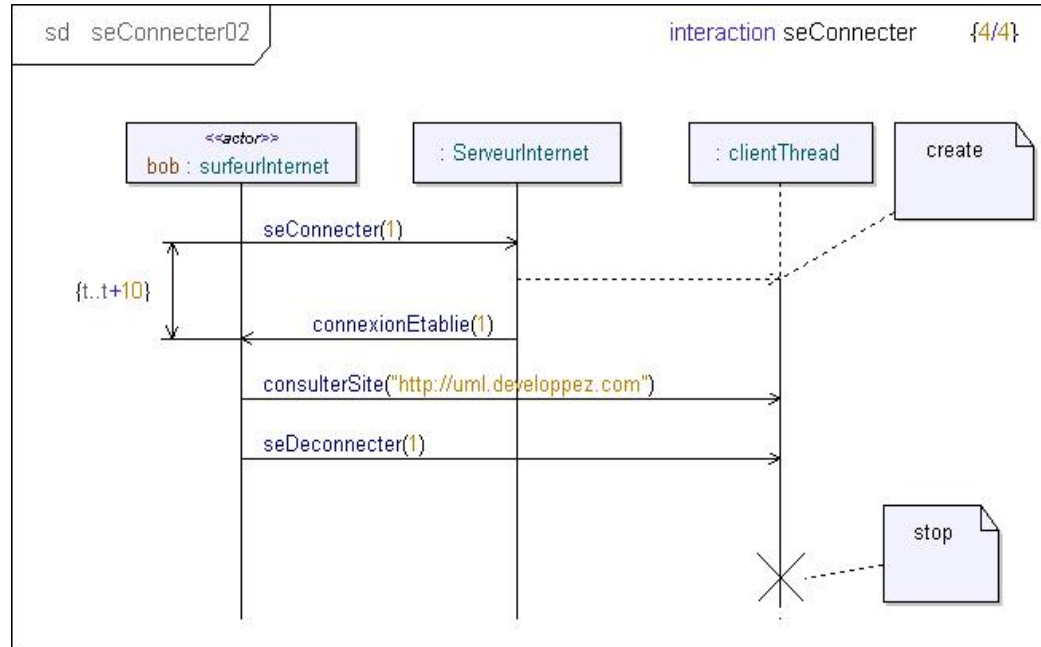
# Diagramme de Séquence

## Contraintes temporelles

- ❑ Des contraintes temporelles peuvent être indiquées sur les diagrammes de séquence.

### Exemple:

Impose une contrainte de temps pour la connexion d'un utilisateur.



# Diagramme de Séquence

## Exemple de la modélisation d'un diagramme de séquence

- ❑ La rubrique « enchaînement nominal » du cas d'utilisation « retrait d'espèces » contient les éléments suivants :
  - 1) Le guichetier saisit le numéro de compte du client ;
  - 2) L'application valide le compte auprès du système central ;
  - 3) Le guichetier demande un retrait de 1000 dh ;
  - 4) Le système « guichet » interroge le système central pour s'assurer que le compte est suffisamment approvisionné ;
  - 5) Le système central effectue le débit du compte ;
  - 6) En retour, le système notifie au guichetier qu'il peut délivrer le montant demandé.

## Exercice 1

Le déroulement normal d'utilisation d'une caisse de supermarché est le suivant :

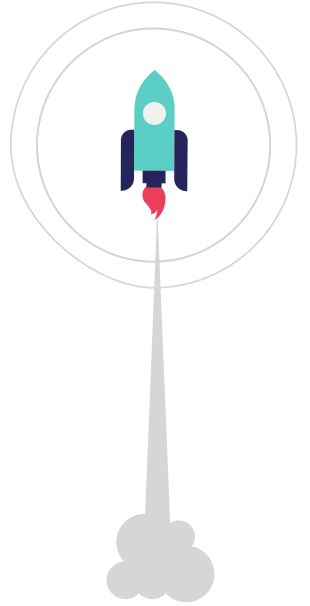
- ☐ un client arrive à la caisse avec ses articles à payer.
- ☐ le caissier enregistre le numéro d'identification de chaque article, ainsi que la quantité.
- ☐ la caisse affiche le prix de chaque article et son libellé.
- ☐ lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente et la caisse affiche le total des achats.
- ☐ le caissier annonce au client le montant total à payer.
- ☐ le client choisit son mode de paiement:
  - ☐ liquide : le caissier encaisse l'argent, la caisse indique le montant à rendre au Client.
  - ☐ chèque : le caissier note le numéro de pièce d'identité du client
  - ☐ carte de crédit : la demande d'autorisation est envoyée avant la saisie
- ☐ la caisse enregistre la vente et l'imprime
- ☐ le caissier donne le ticket de caisse au client

Réalisez Diagramme de séquence en ne prenant en compte que le cas de paiement par liquide



# **Vision Dynamique du Système :**

## **Diagramme d'États-Transitions**



# Diagramme d'États-Transitions

## Introduction

- ❑ En UML, le diagramme d'états permet de modéliser avec précision des comportements et les cycles de vie complexes d'un objet.
- ❑ Le diagramme d'états-transitions (ou simplement d'états) permet de décrire les changements d'états d'un objet (ou d'un composant ou d'un système pris dans son ensemble), en réponse à des évènements.

# Diagramme d'États-Transitions

## Introduction

❑ Pourquoi ?

**Certaines classes du modèle statique** requièrent un diagramme d'états pour comprendre son **comportement dynamique complexe**.



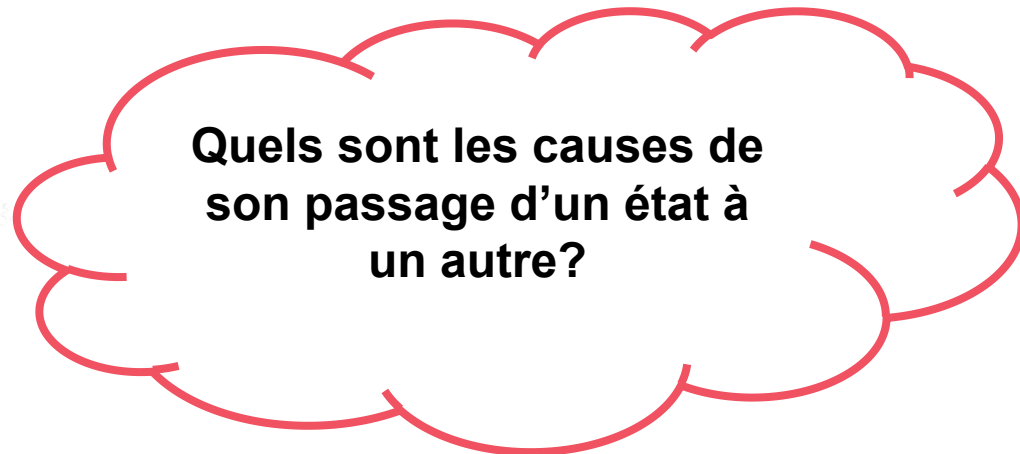
Quels sont les états qu'elle  
peut prendre au cours de sa  
vie?

# Diagramme d'États-Transitions

## Introduction

❑ Pourquoi ?

**Certaines classes du modèle statique** requièrent un diagramme d'états pour comprendre son **comportement dynamique complexe**.



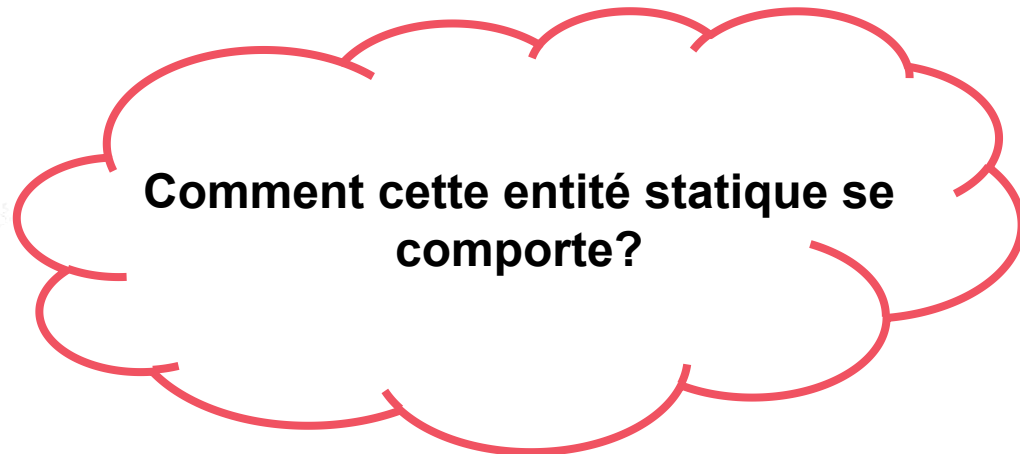
Quels sont les causes de  
son passage d'un état à  
un autre?

# Diagramme d'États-Transitions

## Introduction

❑ Pourquoi ?

**Certaines classes du modèle statique** requièrent un diagramme d'états pour comprendre son **comportement dynamique complexe**.



**Comment cette entité statique se comporte?**

# Diagramme d'États-Transitions

## Introduction

❑ Comment ?

### 1) Choix de l'objet:

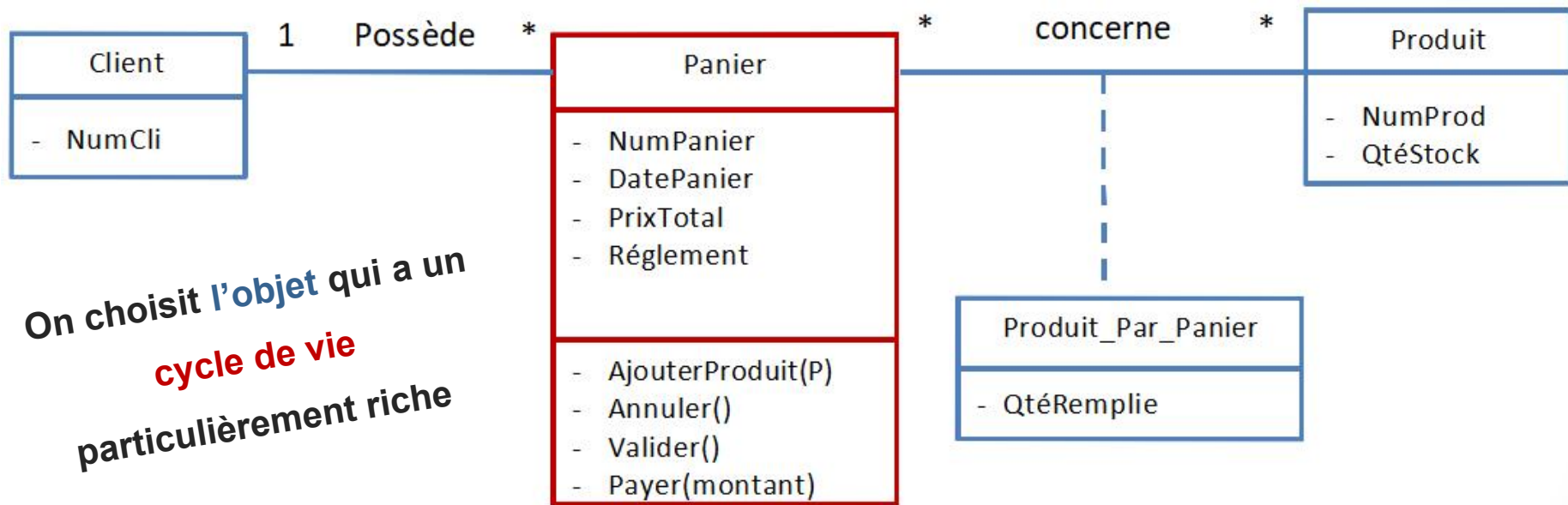
- ❑ Pour réaliser un diagramme d'état, il ne faut pas choisir un objet qui n'a pas d'états ou qui ne change pas d'états.
- ❑ On choisit les objets qui ont un cycle de vie particulièrement riche.

# Diagramme d'États-Transitions

## Introduction

❑ Comment ?

### 1) Choix de l'objet:



On choisit l'objet qui a un  
**cycle de vie**  
particulièrement riche

# Diagramme d'États-Transitions

## Introduction

❑ Comment ? **2) Déterminer les états à modéliser:** L'objet qui nous intéresse c'est le **panier**

Éléments déclencheurs de l'état	États de l'objet
Le panier peut être créé par un client inscrit	Créé
le panier devient susceptible à une opération d'achat probable	En attente
Lorsqu'il ajoute un produit ou retire un produit, le panier devient modifié	Modifié
Lorsqu'il vide le panier, il devient vidé	Vidé
Lorsque le client parcourt les produits du panier, il est consulté	Consulté
Lorsque le client valide les produits du panier, il est validé	Validé
Lorsque le client procède au paiement en ligne, le panier est payé	Payé
Lorsqu'une facture est envoyée au client, il devient facturé	Facturé
Lorsque le client annule le panier, alors celui-ci est détruit	Détruit

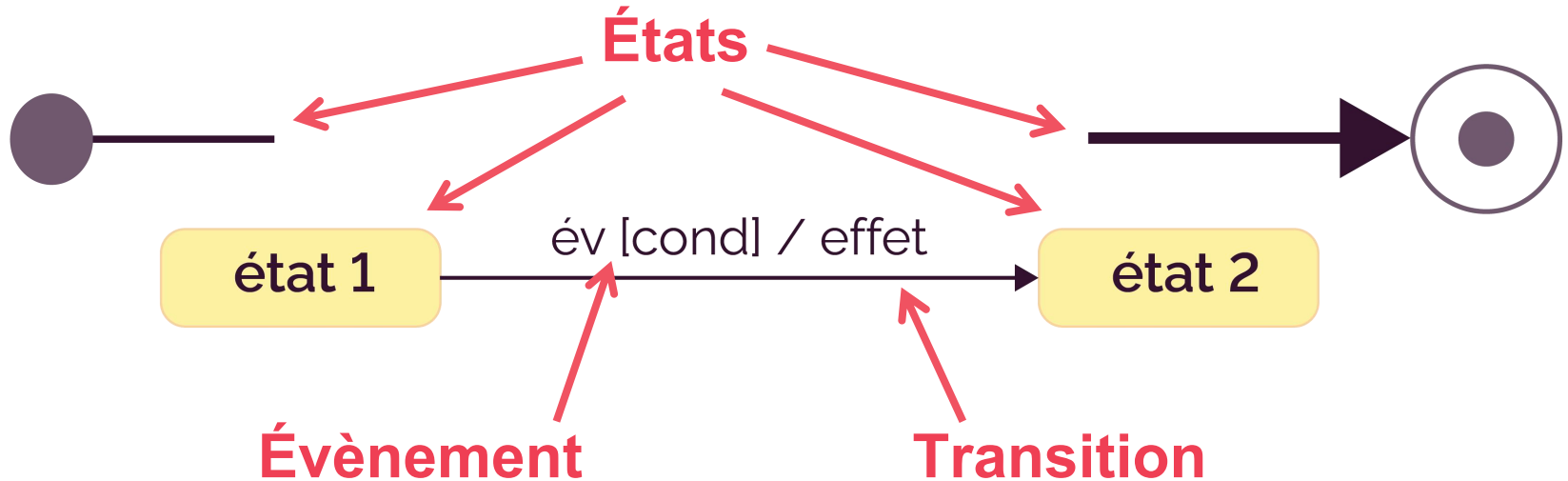


# Diagramme d'États-Transitions

## Introduction

❑ Comment ?

### 3) Conception des états de l'objet:

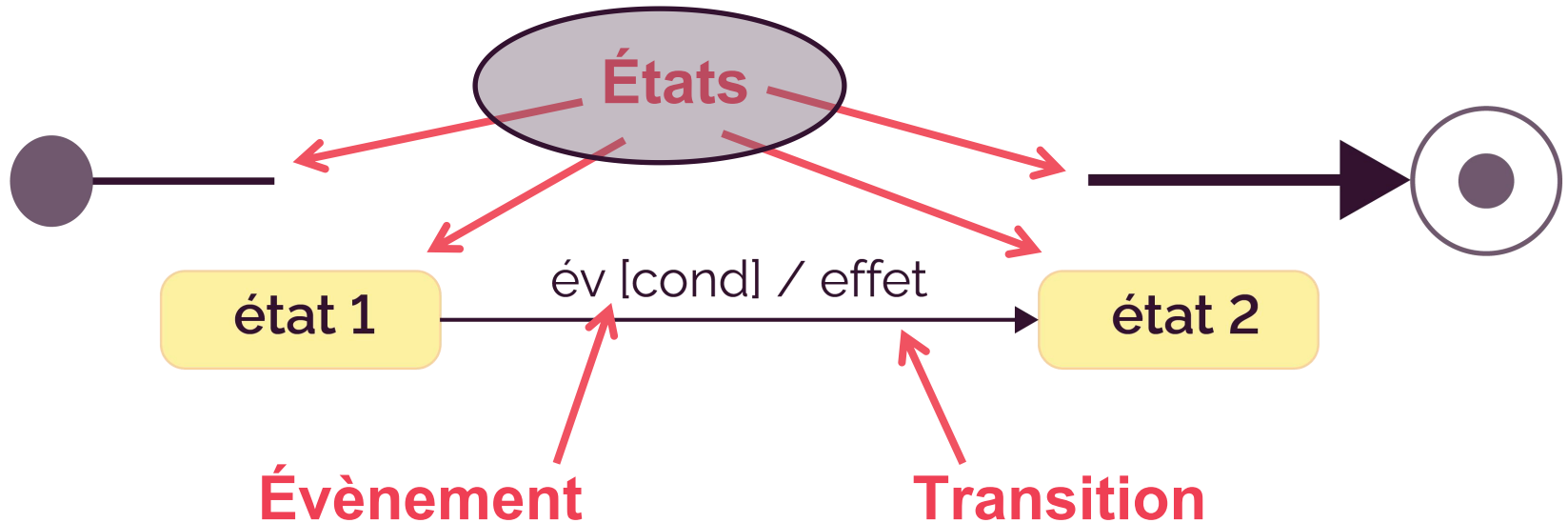


# Diagramme d'États-Transitions

## Introduction

❑ Comment ?

### 3) Conception des états de l'objet:



# Diagramme d'États-Transitions

## États

- ❑ Un état représente une situation durant le vie d'un objet pendant laquelle :  
il exécute une certaine activité ou bien, il attend un certain événement.
- ❑ Un objet passe par une succession d'états durant son existence.
- ❑ Un état a une durée finie.

# Diagramme d'États-Transitions

## États: Trois Types

- ❑ **L'état initial** est l'état dans lequel l'objet se trouve lors de sa création.
- ❑ **L'état final** correspond à la destruction de l'objet.
- ❑ Un diagramme d'états a toujours un et un seul état initial. Il peut n'avoir aucun état final ou plusieurs.
- ❑ Après sa création, un objet passe par une série **d'états normaux**.



État initial



État final

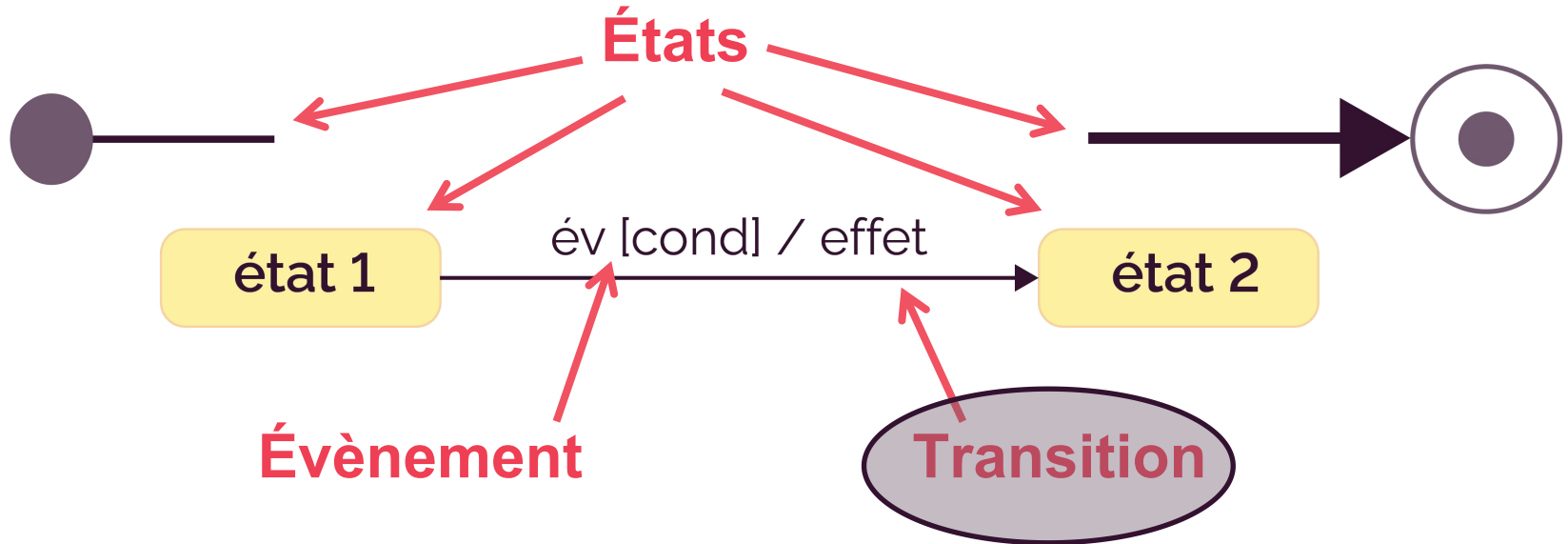
Etat intermédiaire

# Diagramme d'États-Transitions

## Introduction

❑ Comment ?

### 3) Conception des états de l'objet:



# Diagramme d'États-Transitions

## Transition

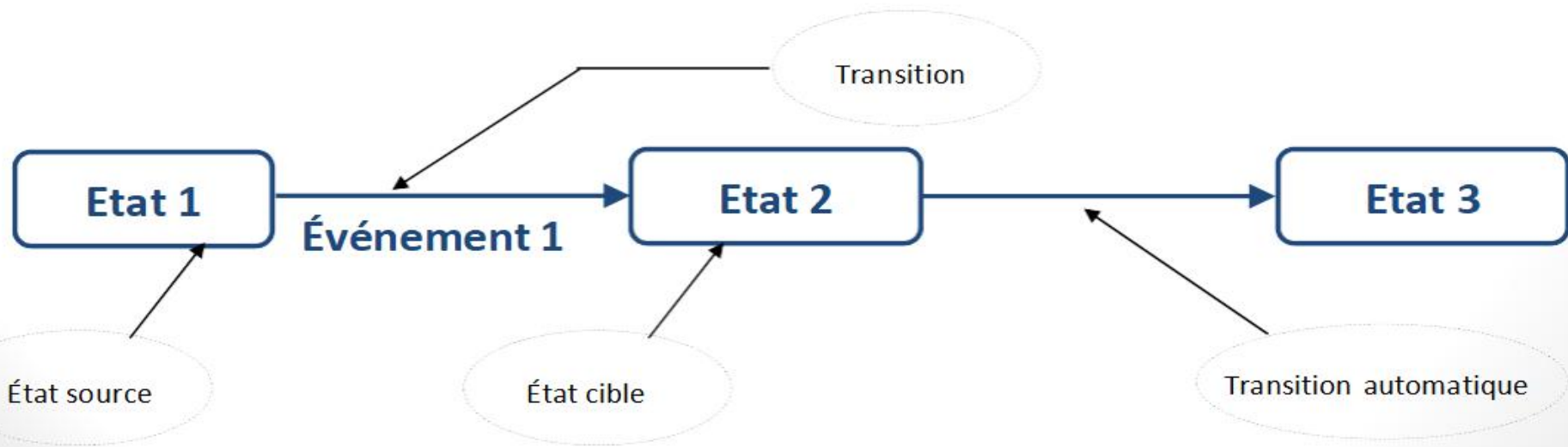
- ❑ Une transition décrit la réponse d'un objet lorsqu'un événement se produit provoquant le passage de l'objet d'un état (état source) dans un autre (état cible).
- ❑ La transition est représentée par une flèche orientée de l'état source vers l'état cible.



# Diagramme d'États-Transitions

## Transition: Deux Types

- ❑ L'événement qui détermine le franchissement de la transition est indiqué sous forme de texte (**Transition par un Évènement**).
- ❑ Si aucun événement n'est spécifié, alors il s'agit d'une **Transition Automatique**.

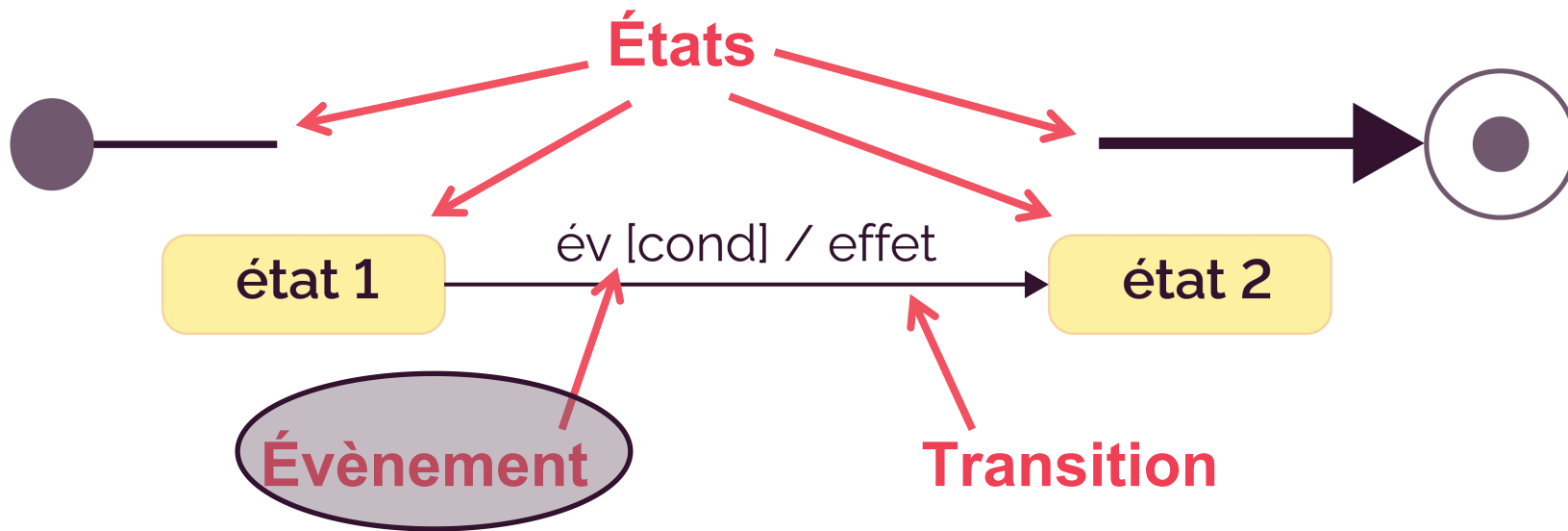


# Diagramme d'États-Transitions

## Introduction

❑ Comment ?

### 3) Conception des états de l'objet:





# Diagramme d'États-Transitions

## Évènement

- ❑ Un événement est un stimulus qui arrive à un moment précis et pouvant déclencher une transition entre états.
- ❑ La syntaxe de la structure de l'événement correspond aux détails de la transition.



# Diagramme d'États-Transitions

## Évènement: Deux Types

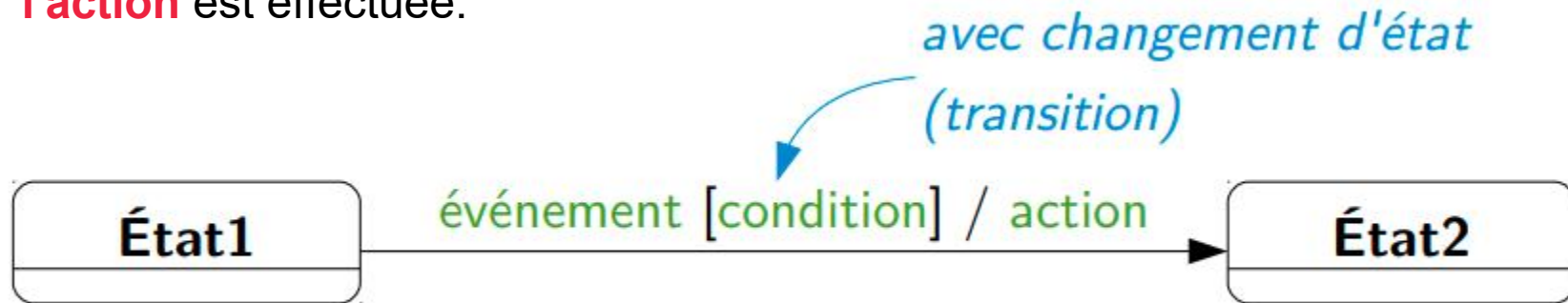
- ❑ **Externe à l'état** : qui change l'état de l'objet.
- ❑ **Interne à l'état** : qui ne change pas l'état de l'objet.



# Diagramme d'États-Transitions

## Évènement: Évènement Externe

- ❑ Lorsque **l'évènement** se produit, si **la garde (Condition)** est vérifiée, alors **l'action** est effectuée.



**Évènement** : déclencheur

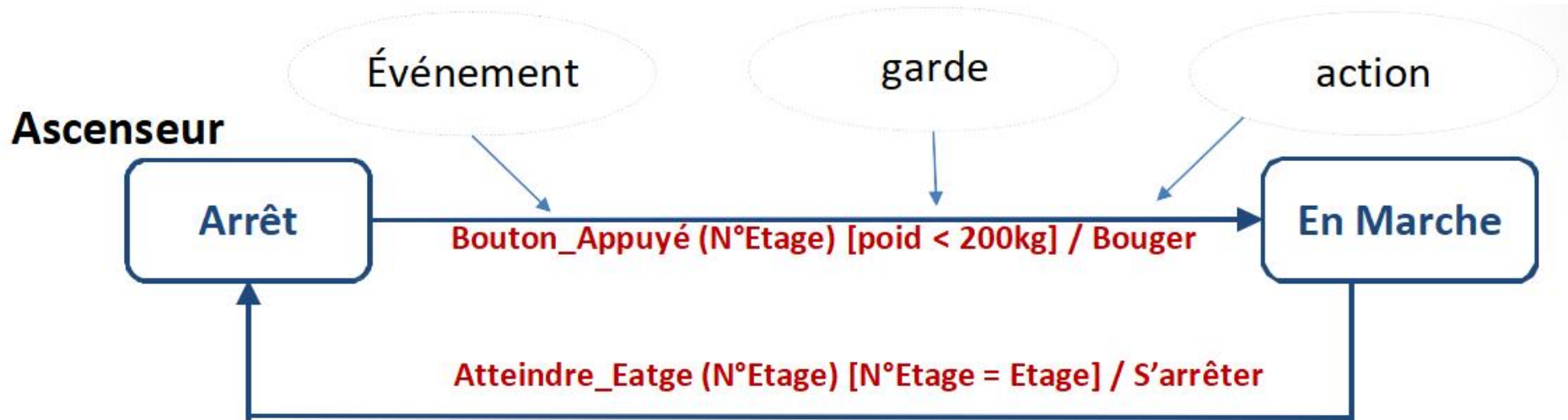
**Garde** : condition avec valeur logique (Vrai ou Faux)

**Action** : spécifie une activité à effectuer lors du déclenchement.

# Diagramme d'États-Transitions

Évènement: Évènement Externe

❑ Exemple:



# Diagramme d'États-Transitions

## Évènement: Évènement Externe

### ❑ Types:

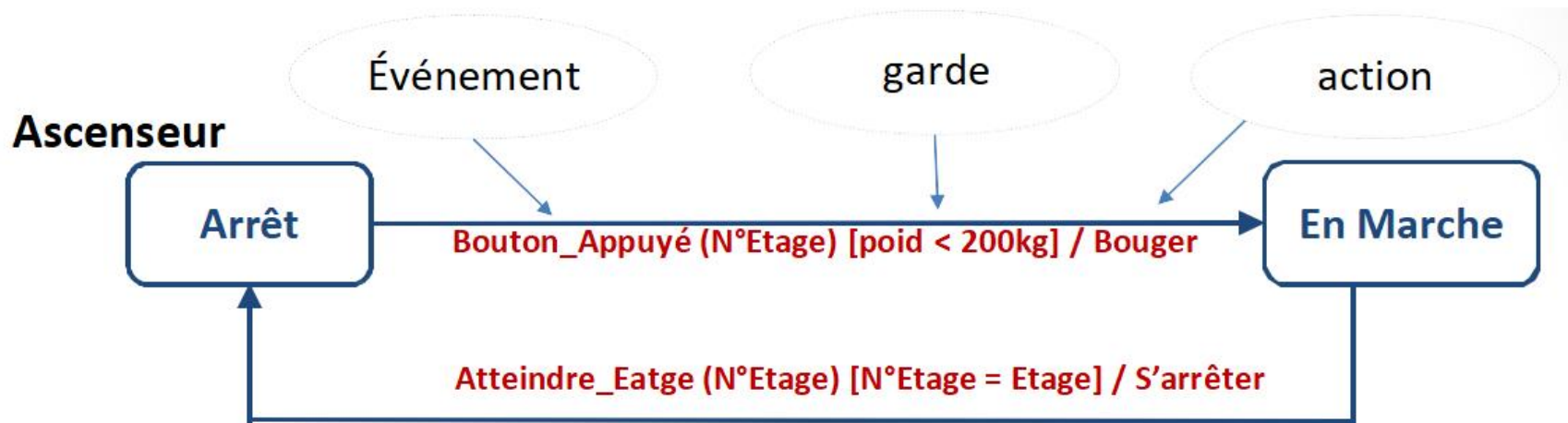
- ❑ **Évènement signal:** Correspond à la réception d'un signal asynchrone émit par un autre objet ou par un acteur.
- ❑ **Évènement appel d'opération** : Appel d'une méthode de l'objet courant par un autre objet ou par un acteur. Il s'agit d'un message synchrone.
- ❑ **Évènement de changement** : Se produit lorsqu'une condition passe de faux à vrai. (modélisé par le mot clé when suivi d'une expression booléenne)
- ❑ **Évènement temporel** : Causé par l'expiration d'une temporisation. (modélisé par le mot clé after suivi d'un expression représentant une durée)

# Diagramme d'États-Transitions

## Évènement: Évènement Externe

□ Types:

- **Évènement signal:** Correspond à la réception d'un signal asynchrone émit par un autre objet ou par un acteur.

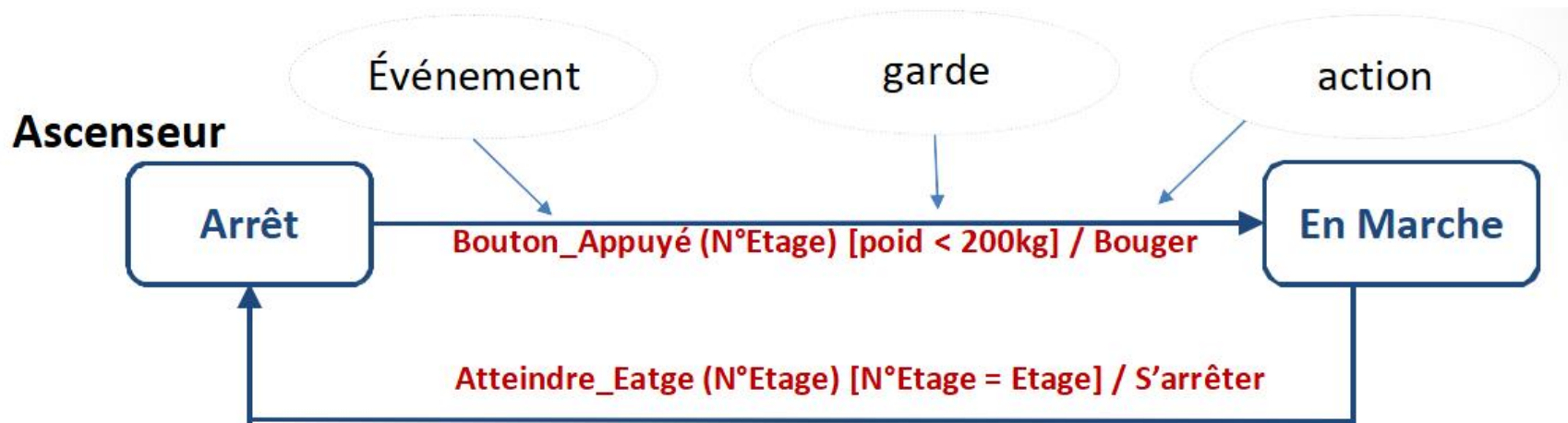


# Diagramme d'États-Transitions

## Évènement: Évènement Externe

□ Types:

- **Évènement appel d'opération** : Appel d'une méthode de l'objet courant par un autre objet ou par un acteur. Il s'agit d'un message synchrone.



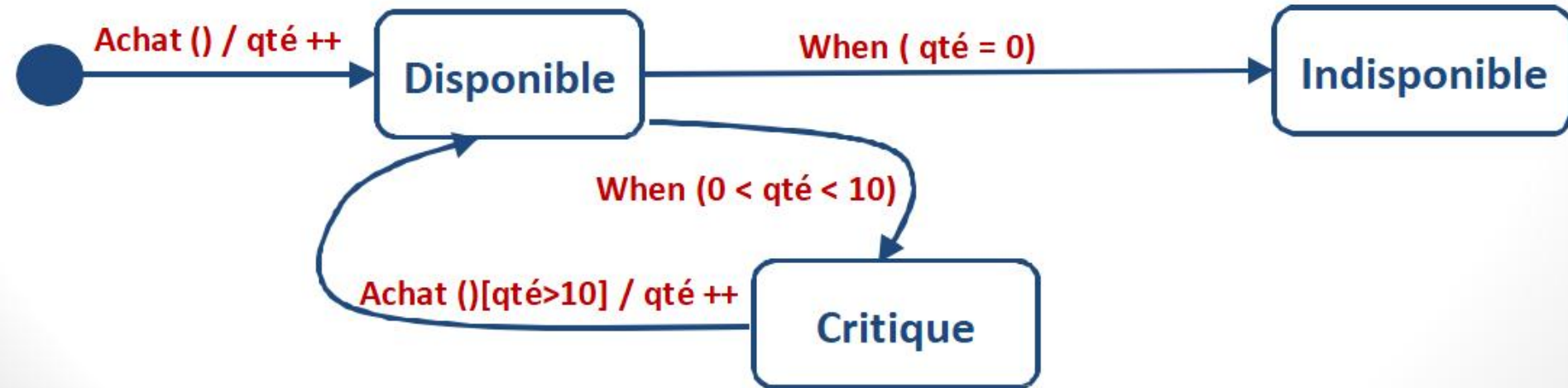
# Diagramme d'États-Transitions

## Évènement: Évènement Externe

❑ Types:

❑ **Évènement de changement** : Se produit lorsqu'une condition passe de faux à vrai.

when ( expression booléenne)





# Diagramme d'États-Transitions

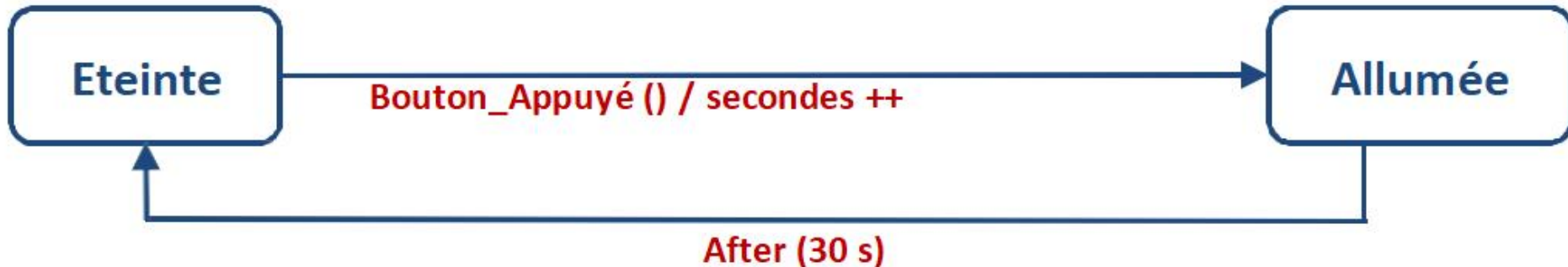
## Évènement: Évènement Externe

❑ Types:

❑ **Évènement temporel** : Causé par l'expiration d'une temporisation.

when ( date = « expression date précise »)

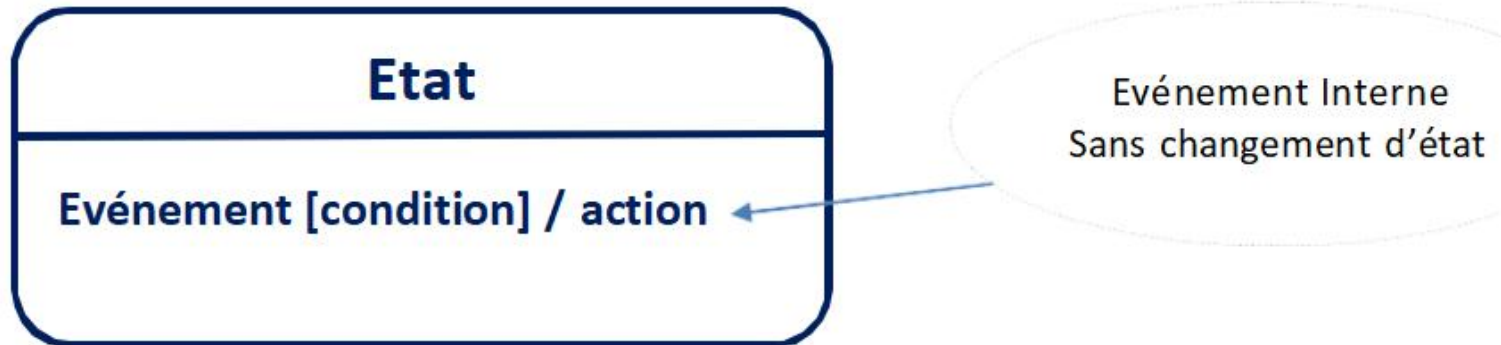
after (« expression d'une durée »)



# Diagramme d'États-Transitions

## Évènement: Évènement Interne

- ❑ On peut avoir des états qui effectuent plusieurs activités successivement ou en parallèle.
- ❑ L'enchaînement de ces activités à l'intérieur d'un même état peut être spécifié grâce aux événements internes qui ont la même syntaxe que les événements externes.



# Diagramme d'États-Transitions

## Évènement: Évènement Interne

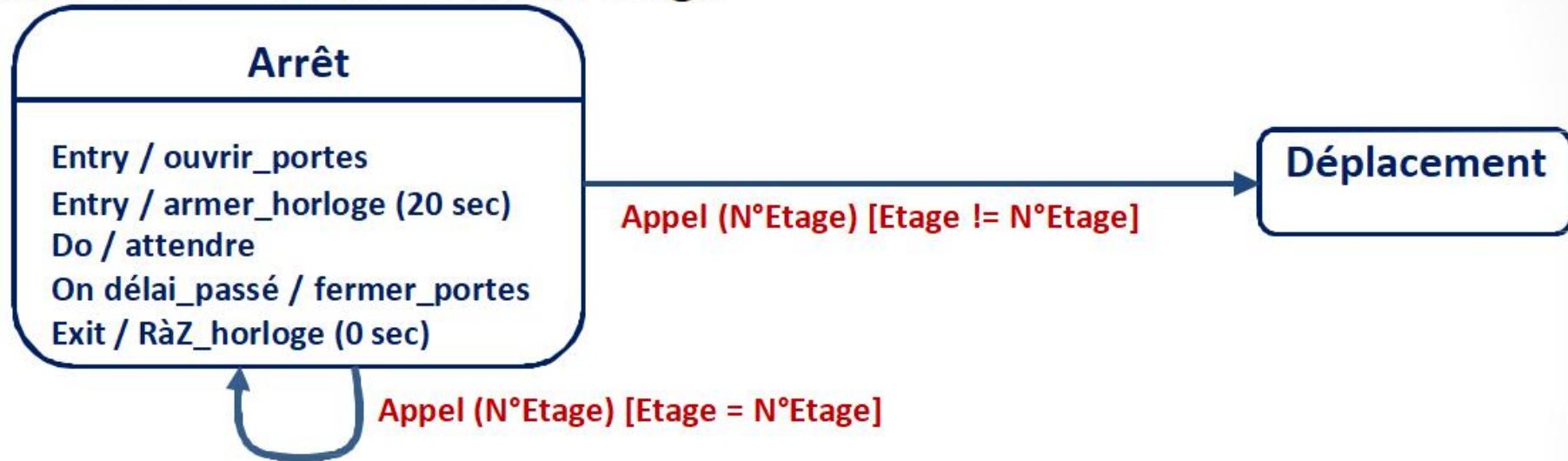
- ❑ UML définit des mots clé correspondant à des événements internes particuliers :
  - ❑ **Entry** / activité entrée : événement à l'entrée dans l'état.
  - ❑ **Do** / activité : définit l'activité à exécuter dès que celle définie par entry est terminée.
  - ❑ **On event** / activité : (optionnel) définit l'activité à exécuter à chaque fois que nous avons un événement particulier.
  - ❑ **Exit** / activité sortie : événement à la sortie de l'état.

# Diagramme d'États-Transitions

## Évènement: Évènement Interne

- ❑ UML définit des mots clé correspondant à des événements internes particuliers :
- ❑ Exemple:

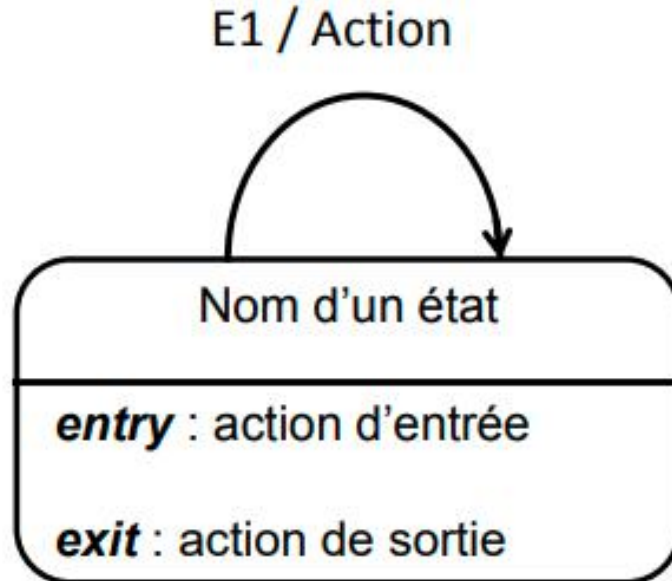
### Etats d'un ascenseur arrêté à l'étage



# Diagramme d'États-Transitions

## La Notion : Transition Réflexive

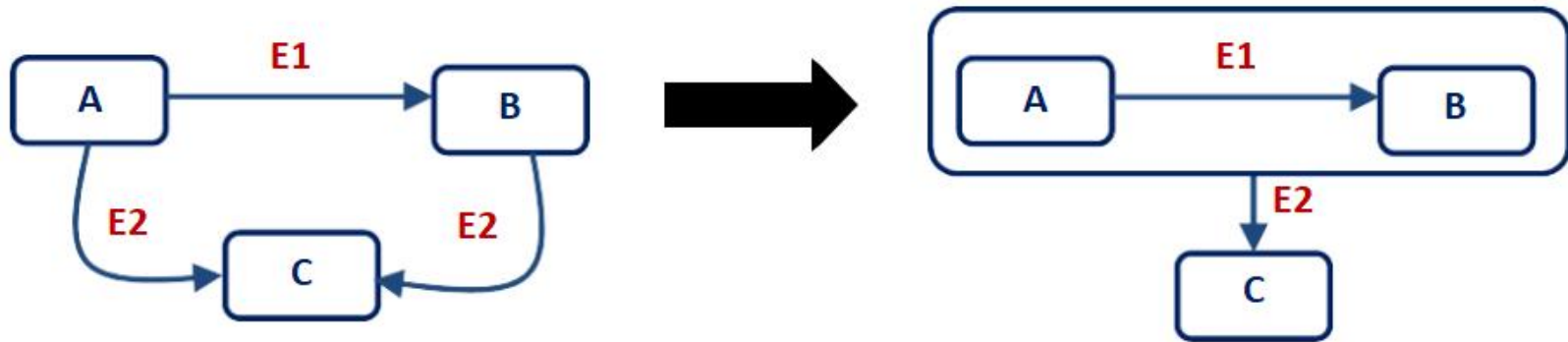
- ❑ Une transition réflexive = cas particulier de transition où état avant-transition = état après-transition



# Diagramme d'États-Transitions

## Notations Avancées: Etat Composite

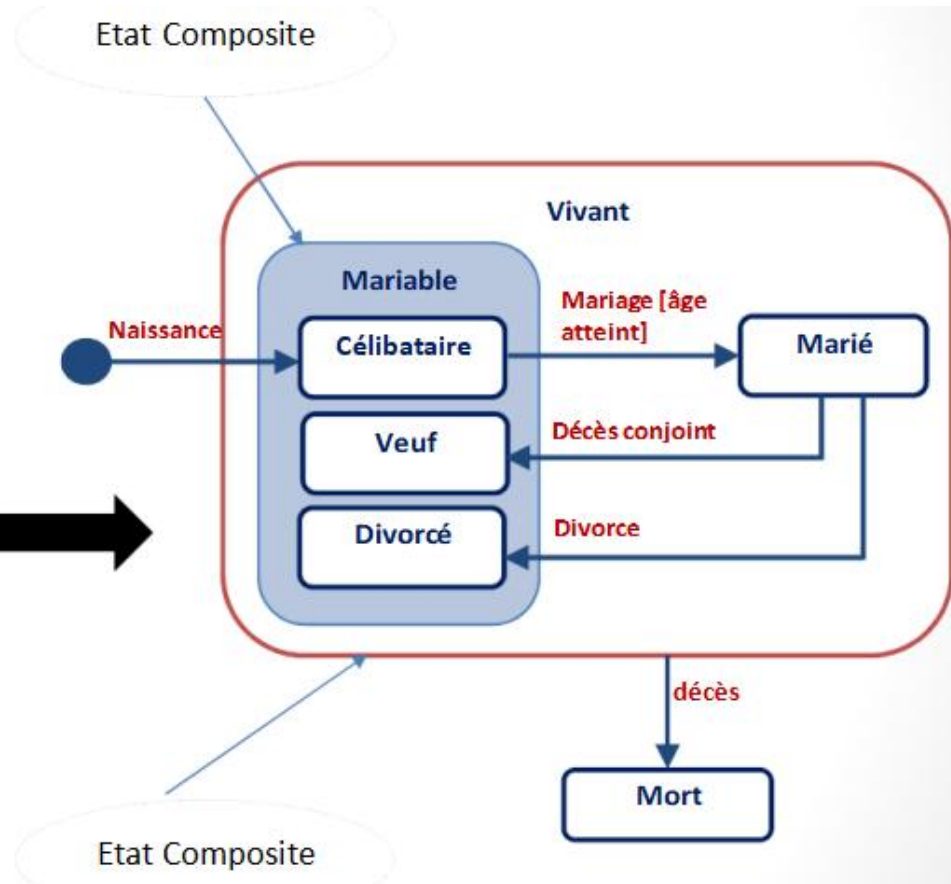
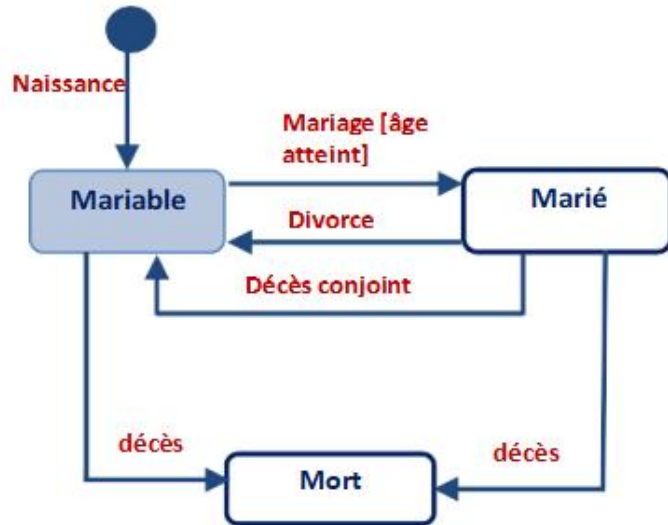
- ❑ Un état composite est un état décomposable en sous états.
- ❑ Les sous-états sont des états composites ou élémentaires.
- ❑ Correspond à une hiérarchisation (généralisation) d'un ensemble d'états.
- ❑ Objectif: Facilite la représentation lorsque le nombre de connexion à un état devient élevé.



# Diagramme d'États-Transitions

## Notations Avancées: Etat Composite

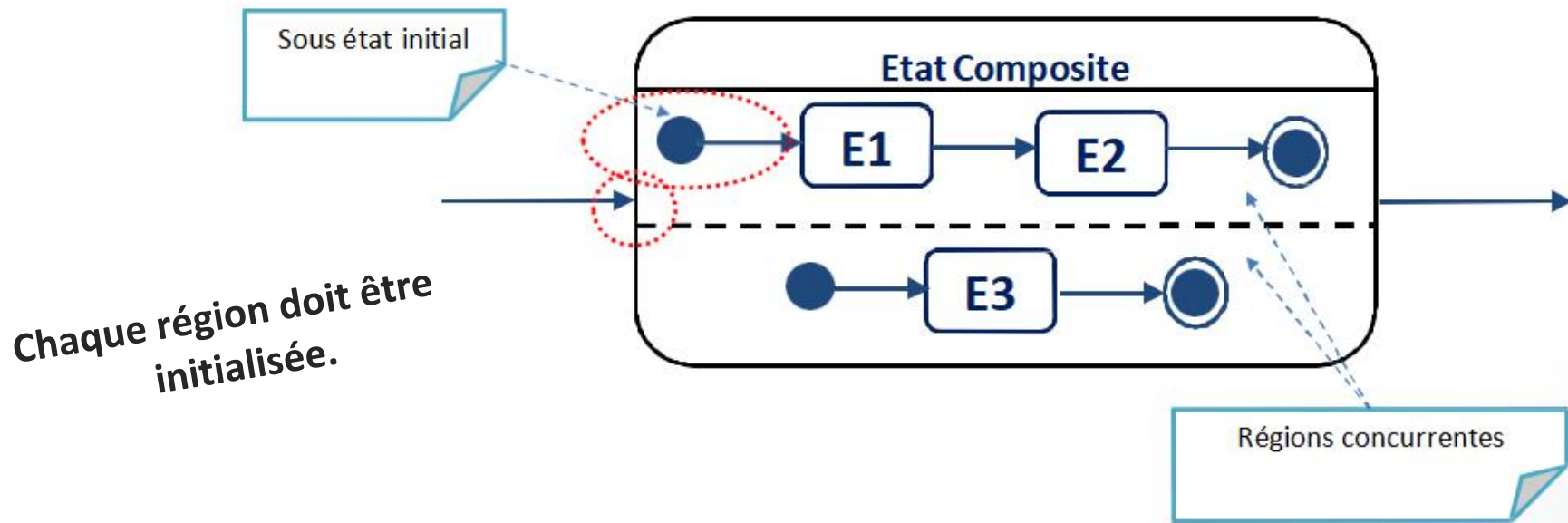
❑ Exemple:



# Diagramme d'États-Transitions

## Notations Avancées: Régions Concurrentes d'un Etat

- ❑ Dans certains cas, plusieurs états peuvent être actifs simultanément.
- ❑ Représentés par des régions parallèles ou concurrentes.

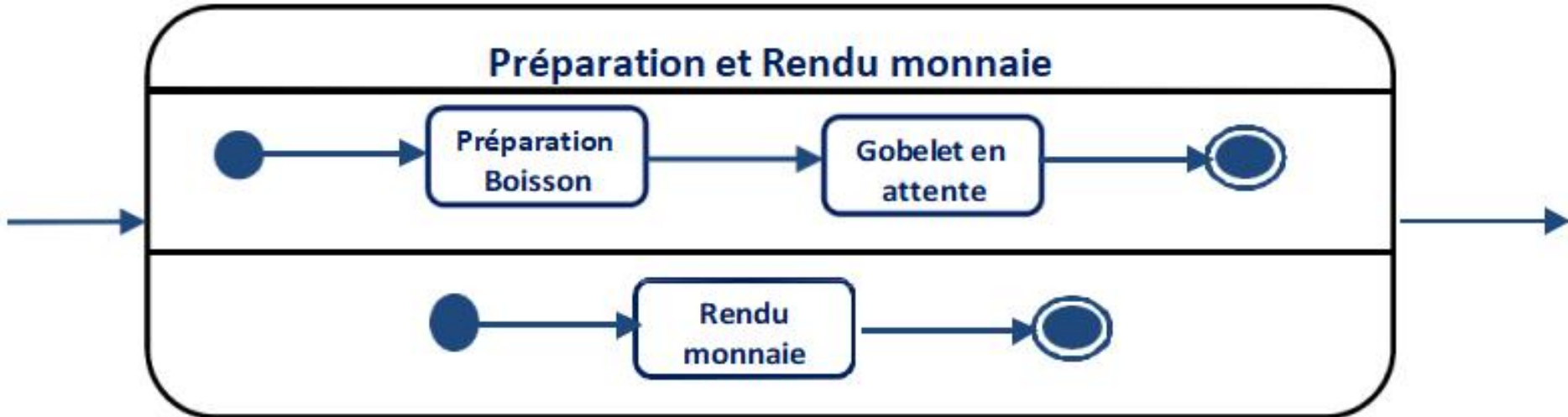




# Diagramme d'États-Transitions

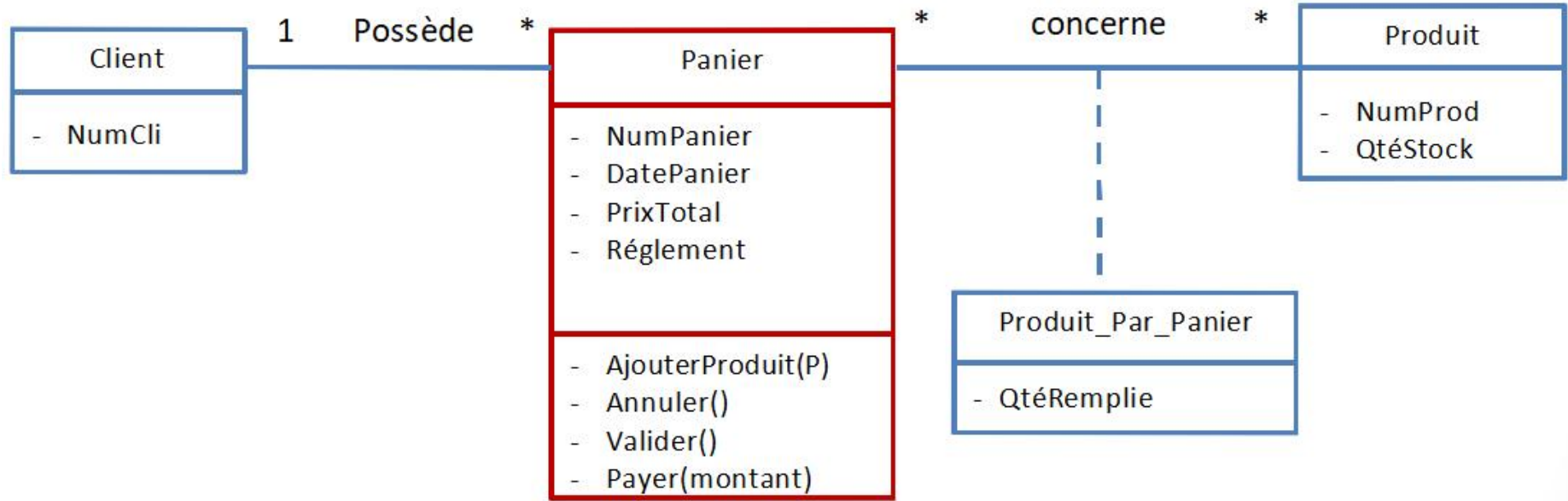
## Notations Avancées: Régions Concurrentes d'un Etat

- ❑ Distributeur automatique de boissons
  - ❑ On prépare la boisson et on rend la monnaie en même temps.



# Exemple de conception d'un diagramme d'États-Transitions

## 1) Choix de l'objet:



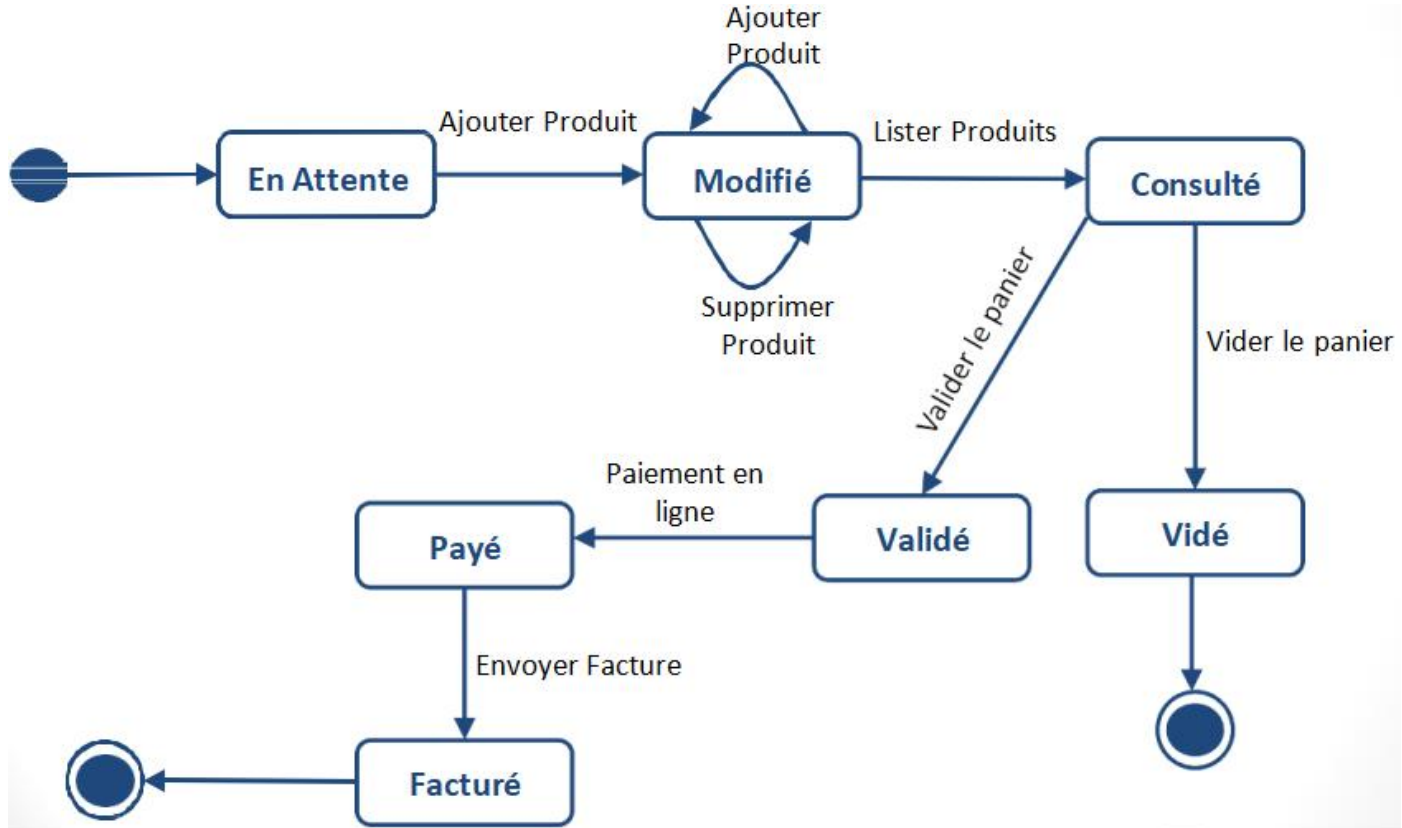
# Exemple de conception d'un diagramme d'États-Transitions

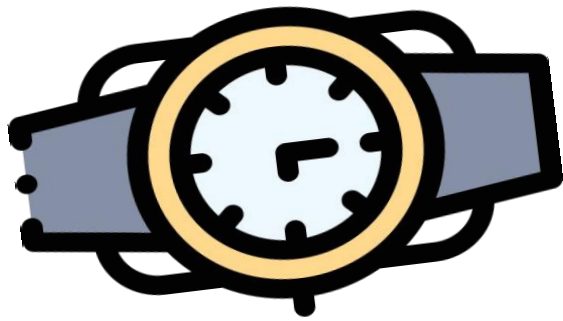
## 2) Déterminer les états à modéliser:

Éléments déclencheurs de l'état	États de l'objet
Le panier peut être créé par un client inscrit	Créé
le panier devient susceptible à une opération d'achat probable	En attente
Lorsqu'il ajoute un produit ou retire un produit, le panier devient modifié	Modifié
Lorsqu'il vide le panier, il devient vidé	Vidé
Lorsque le client parcourt les produits du panier, il est consulté	Consulté
Lorsque le client valide les produits du panier, il est validé	Validé
Lorsque le client procède au paiement en ligne, le panier est payé	Payé
Lorsqu'une facture est envoyée au client, il devient facturé	Facturé
Lorsque le client annule le panier, alors celui-ci est détruit	Détruit

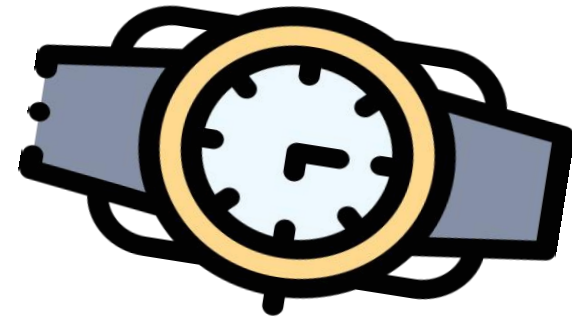
# Exemple de conception d'un diagramme d'États-Transitions

## 2) La conception des états:





## Exercice 1



**Une montre affiche l'heure**, si j'appuie 2X sur le bouton 1, la montre passe en mode “**modificationHeure**”. Chaque pression sur le bouton 2, incrémente l'heure d'une unité. Si j'appuie encore une fois sur le bouton 1, je peux **régler les minutes** de la même façon que les heures. Si j'appuie une quatrième fois sur le bouton 1, la montre **affiche à nouveau l'heure** courante.

## Exercice 2

Un dispositif de contrôle d'accès par carte magnétique à un photocopieur est équipé d'un écran de visualisation qui peut afficher les messages suivants :

**"INSEREZ VOTRE CARTE"** lorsque le dispositif est inutilisé.

**"PATIENTER"** pendant que le dispositif lit le code d'une carte introduite.

**"CARTE INVALIDE"** lorsque le code n'est pas reconnu (illisible) ; la carte est alors automatiquement éjectée.

**"COMPOSEZ VOTRE CODE"** lorsque celui-ci a pu être lu.

**"CODE REFUSE"** si le code composé n'est pas identique au code lu ; la carte est alors automatiquement éjectée.

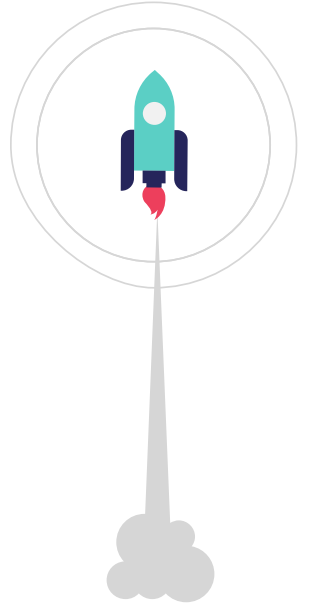
**"UTILISATION EN COURS"** lorsque le code composé est correct.

L'utilisateur peut à tout moment actionner un bouton qui provoque l'éjection de la carte.

Après toute éjection de carte, le dispositif affiche "INSERER CARTE".  
Proposer le graphe états-transitions du lecteur de carte.

# **Vision Dynamique du Système :**

## **Diagramme d'activités**



# Diagramme d'activités

## Introduction

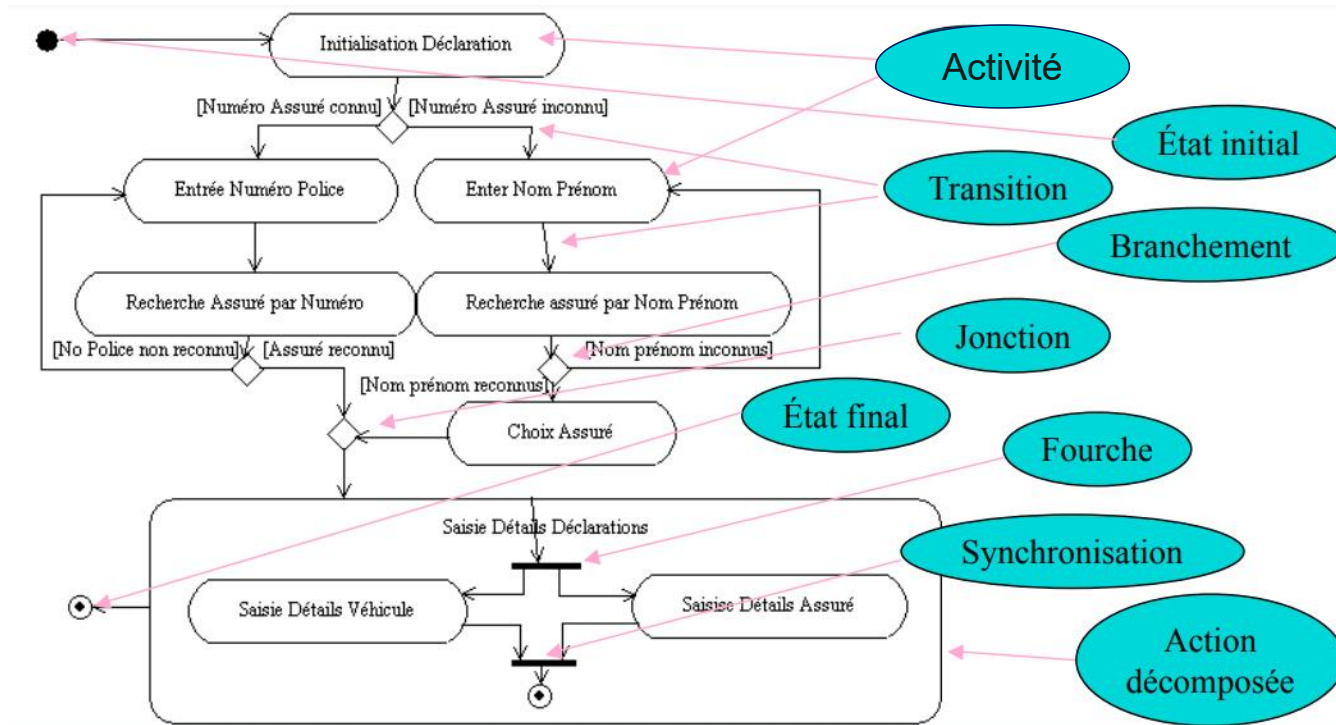
- ❑ Variante des diagrammes d'états-transitions.
- ❑ Le diagramme d'activité permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.
- ❑ Représente le déroulement des traitements en les regroupant dans des étapes appelées « Activité ».

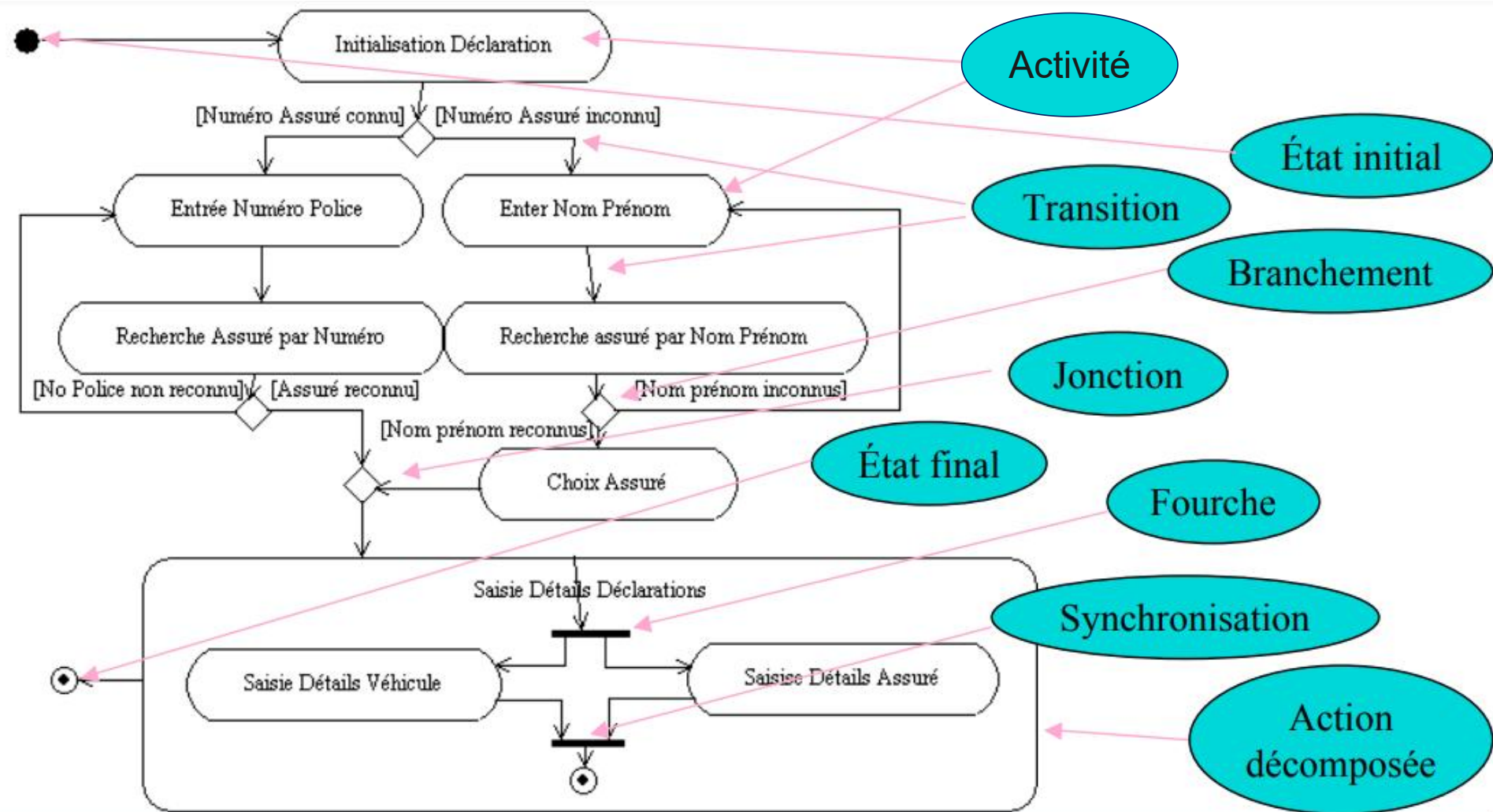


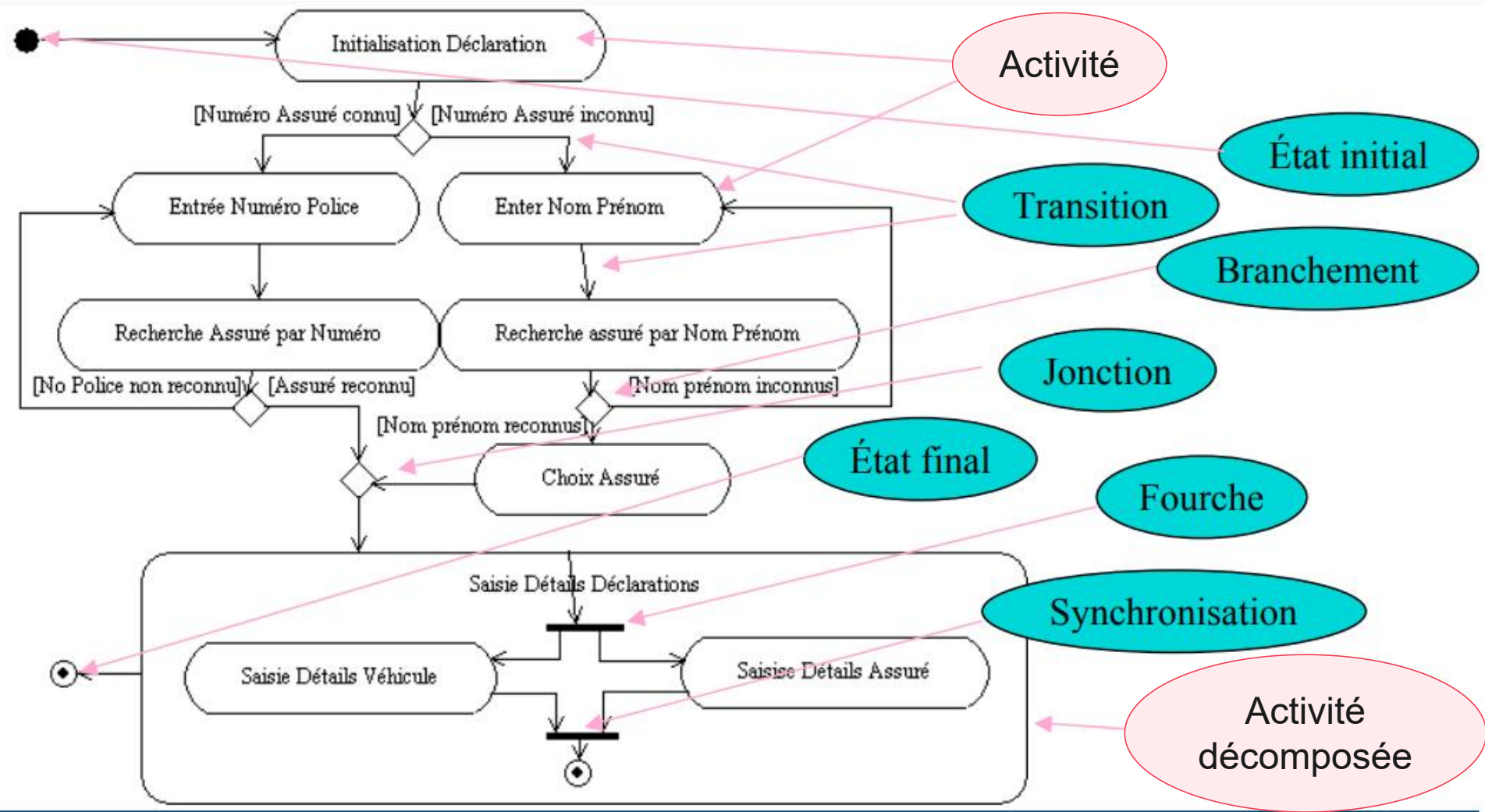
# Diagramme d'activités

## Concepts

❑ Un exemple d'un diagramme d'activité est illustrée ci- dessous:







# Diagramme d'activités

## Activité

❑ Deux types:

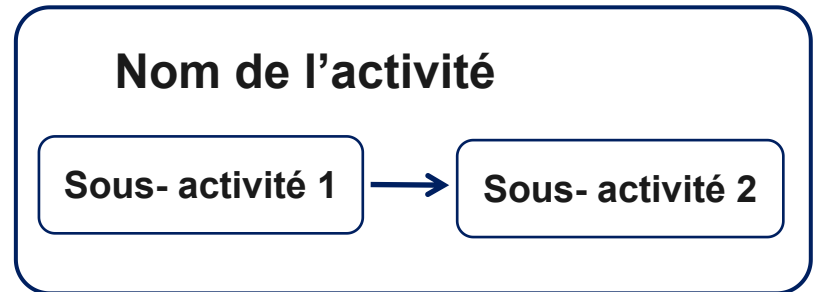
❑ **Activité simple:** Une activité représente l'exécution d'un mécanisme, le déroulement d'un processus.

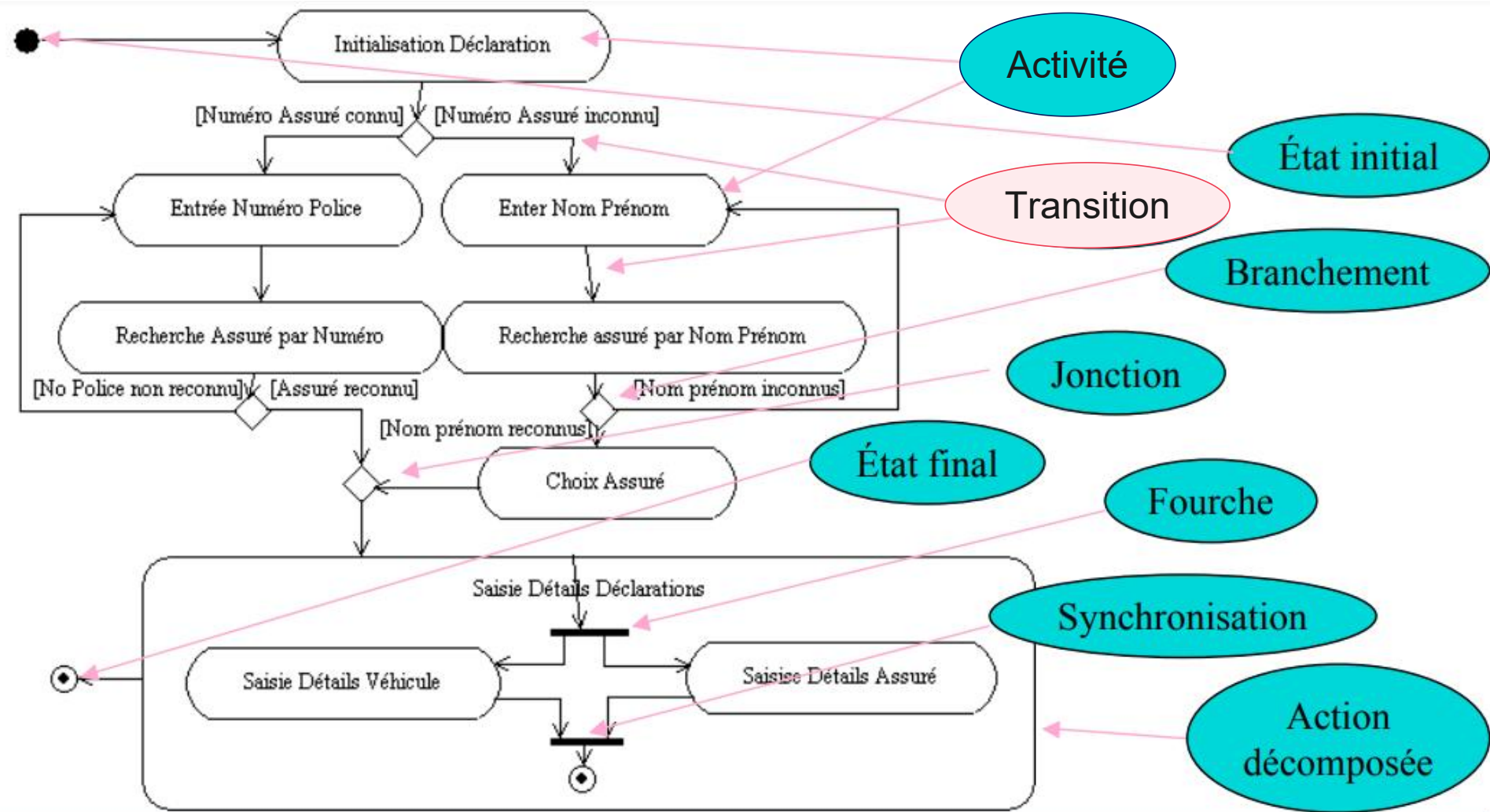
❑ Représentation graphique:



❑ **Activité décomposée:** Une activité décomposée représente un graphe d'activité lui-même.

❑ Représentation graphique:

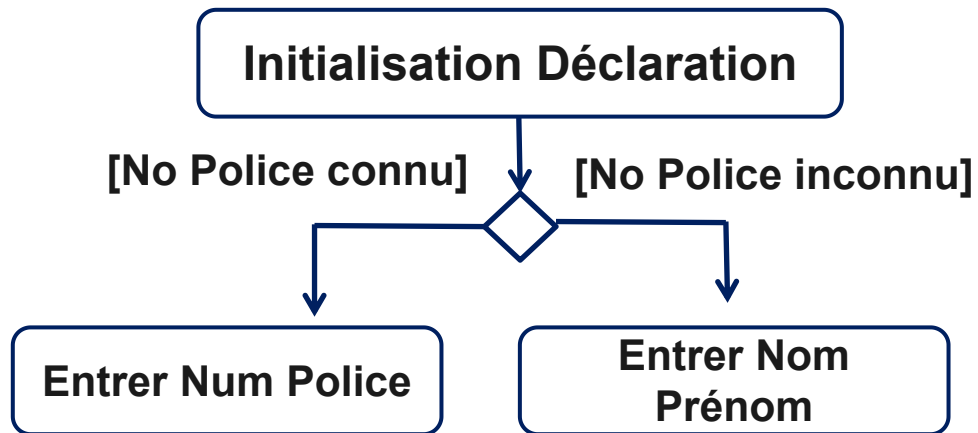
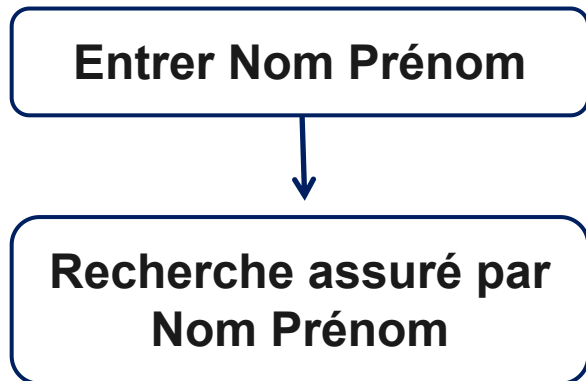




# Diagramme d'activités

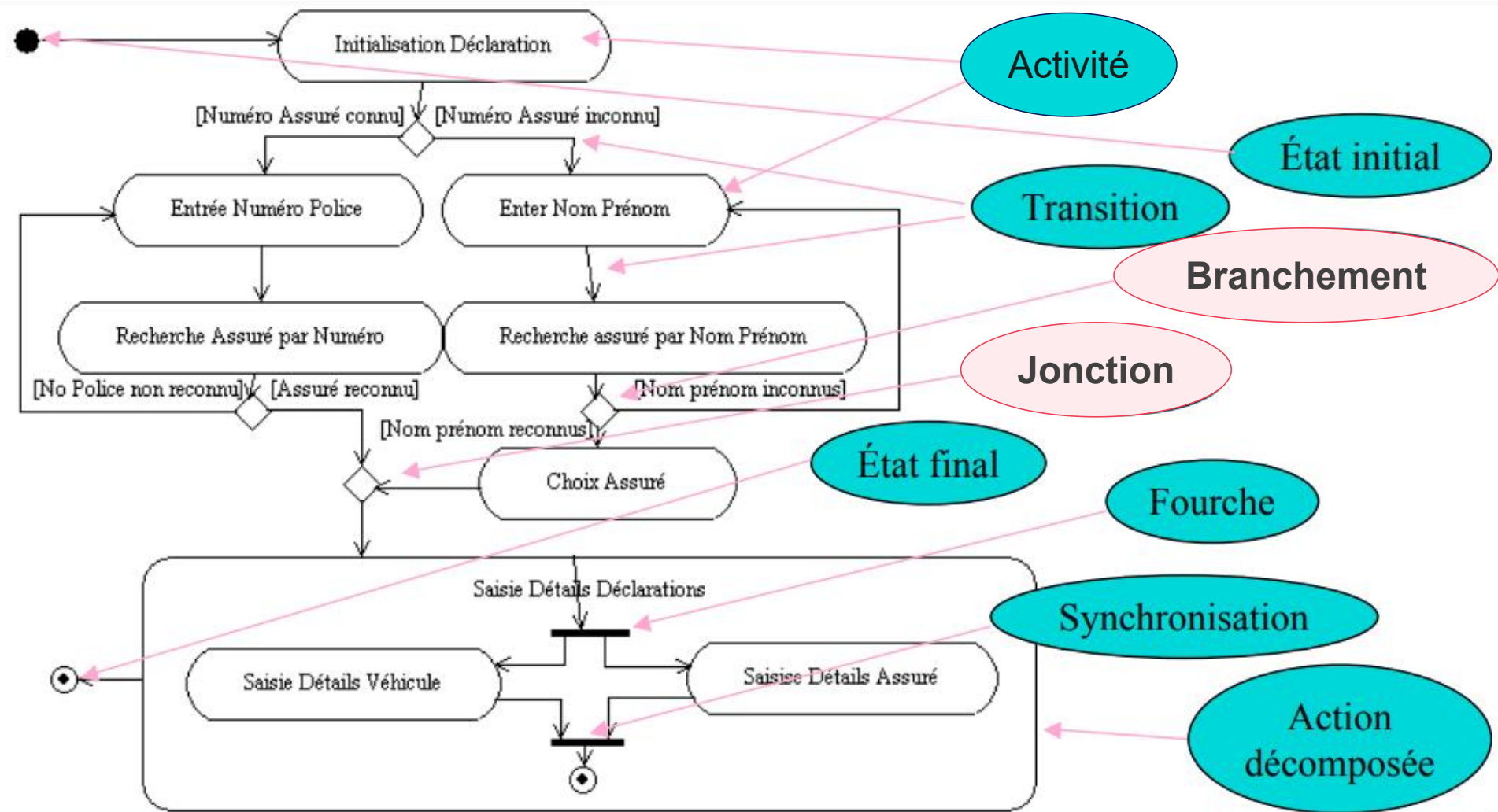
## Transition

- ❑ Deux types:
- ❑ **Transition automatique:** Elle exprime la succession de deux activités.
- ❑ **Transition gardée:** La transition peut être soumise à une garde (condition).
- ❑ Représentation graphique:



- ❑ S'applique surtout entre **un branchement** et des activités

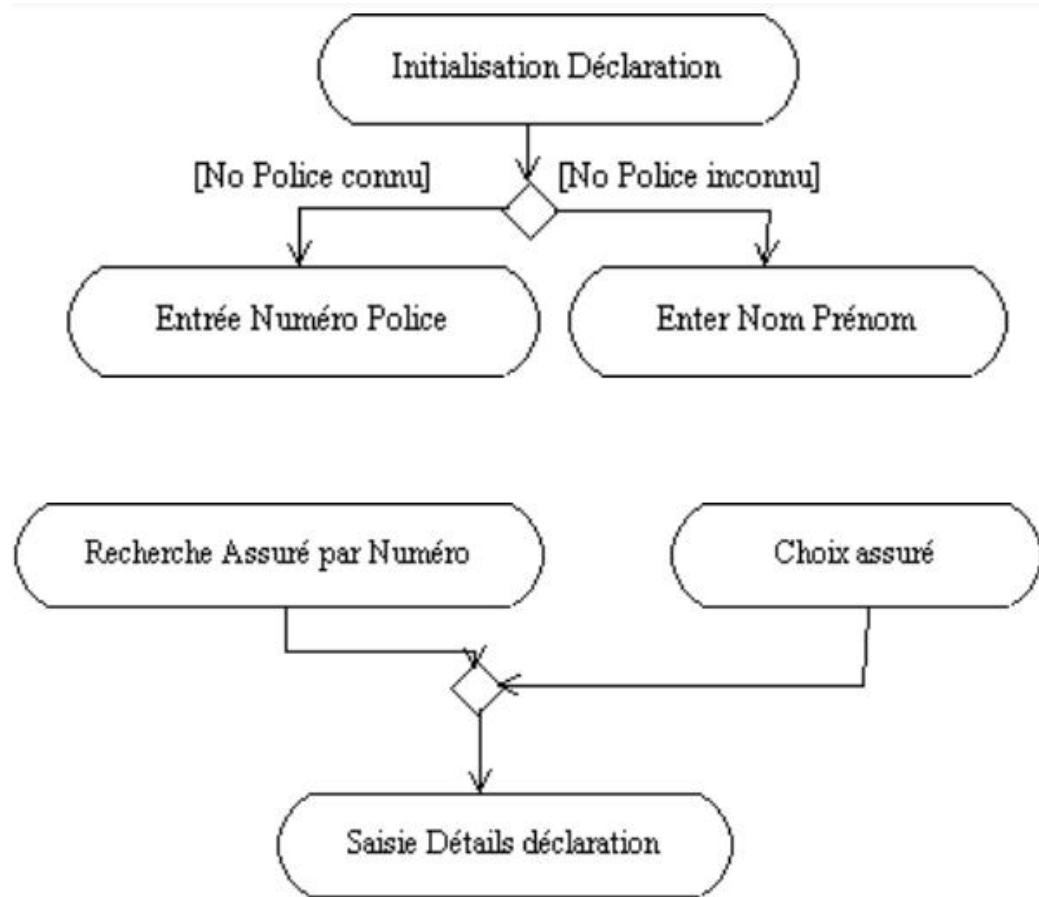




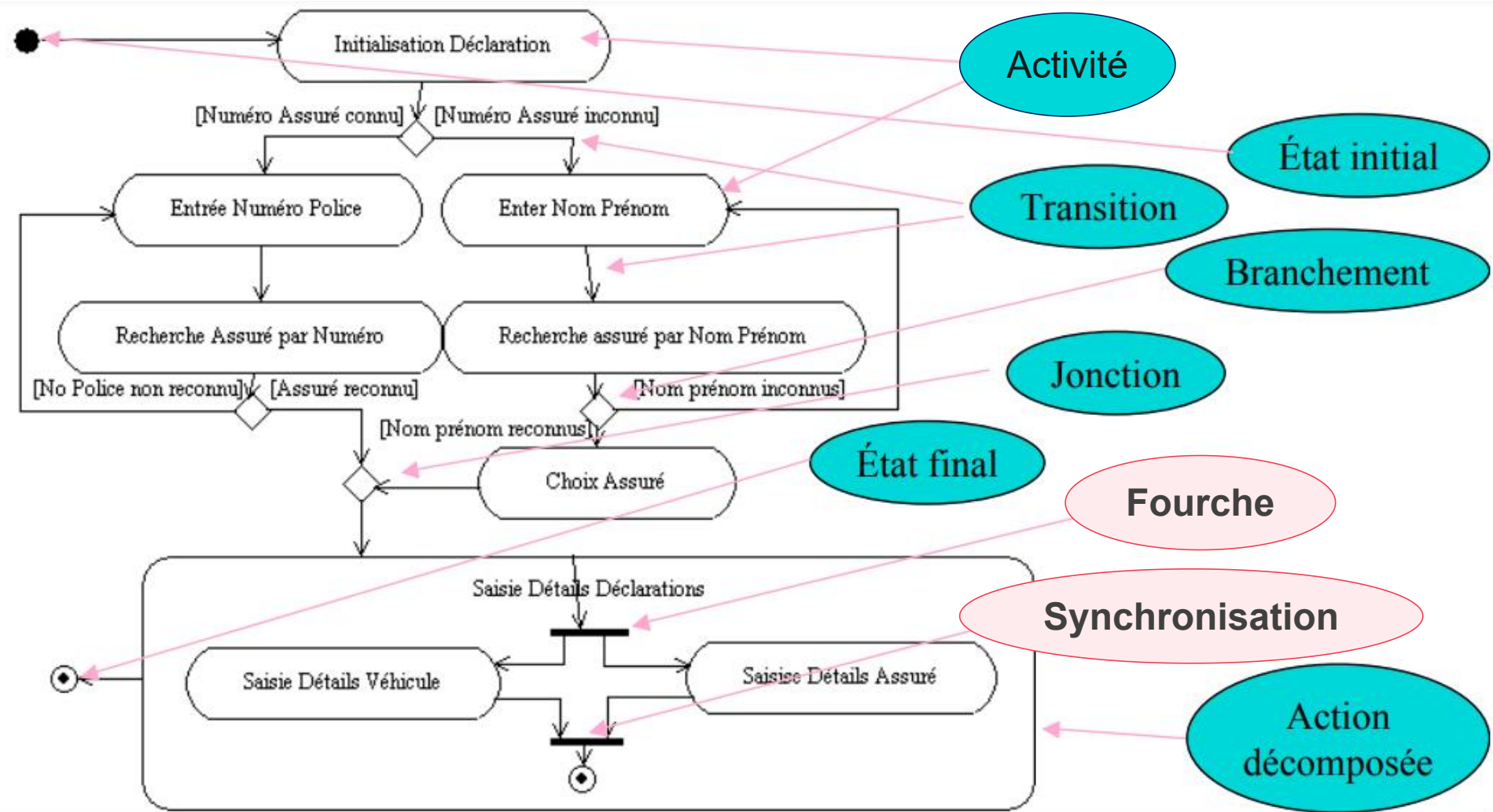
# Diagramme d'activités

## Alternatives

- ❑ **Branchements:** Choix d'un flux d'activité suivant des gardes.
- ❑ Les gardes d'un branchement sont exclusives et totales.
- ❑ **Jonctions:** Regroupement de flux séparés par des branchements.



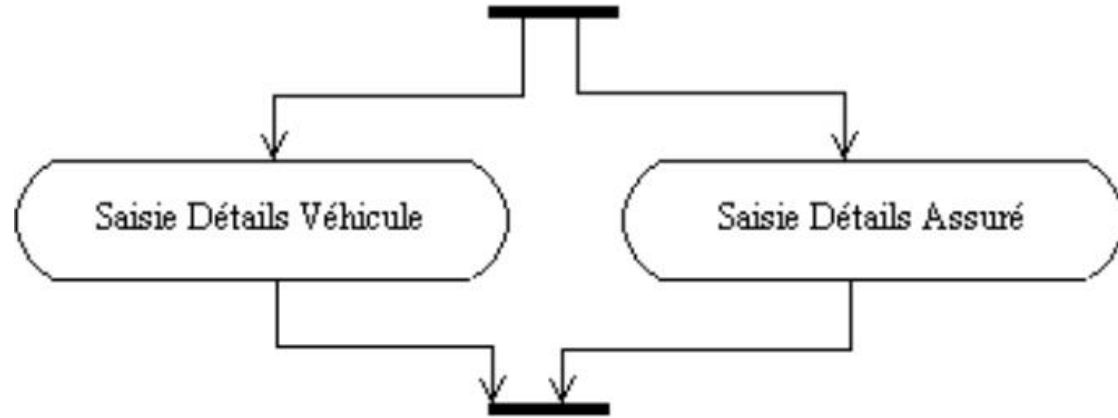




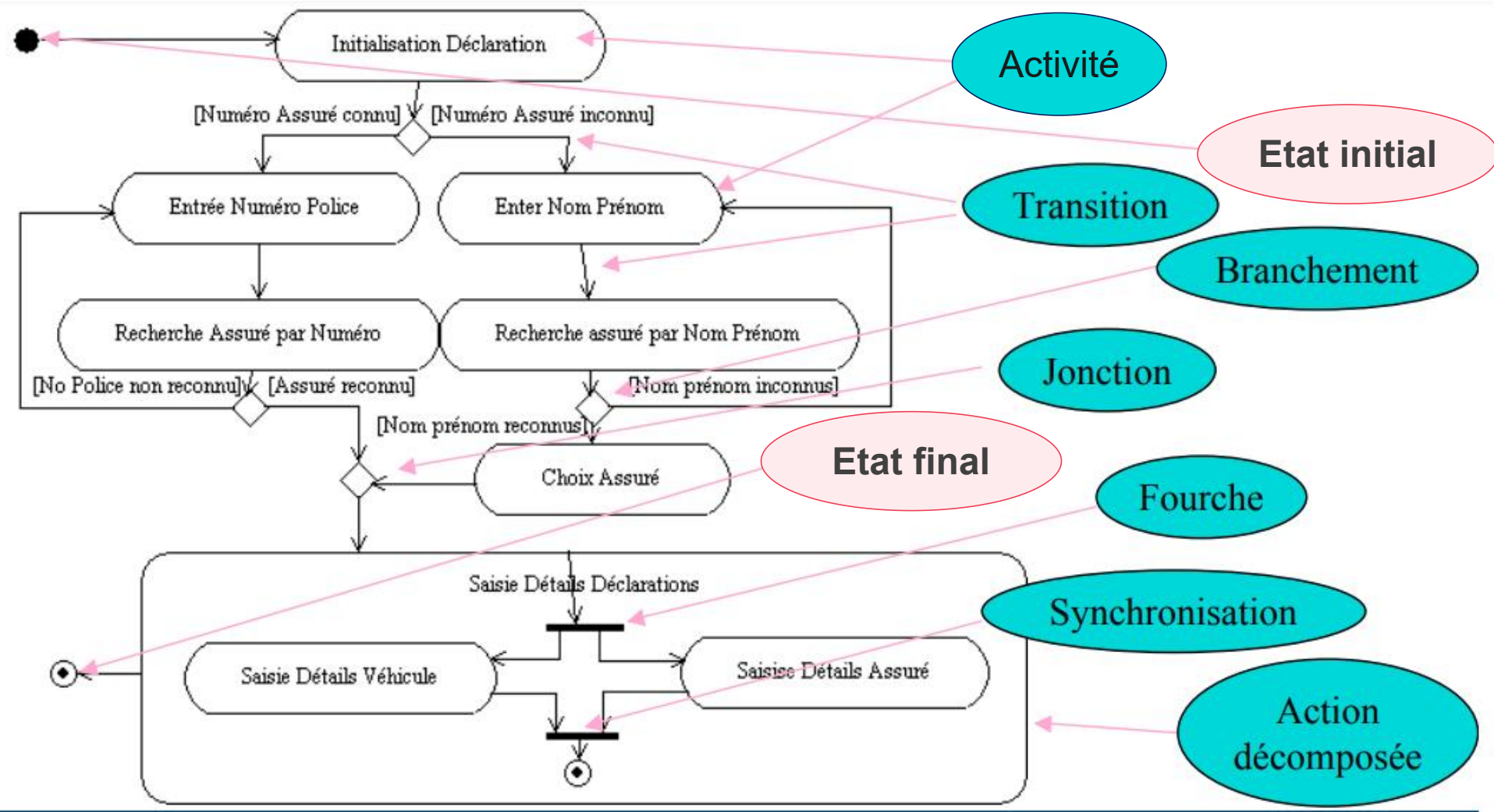
# Diagramme d'activités

## Parallélisme

- ❑ **Fourche:** Lancement d'activités en parallèle.
- ❑ **Synchronisation:** Synchronisation d'activités lancées en parallèle.



**Attention:**  
Tout « Fourche » doit être résolu  
par une « Synchronisation » sous  
peine d'être à jamais parallèle...



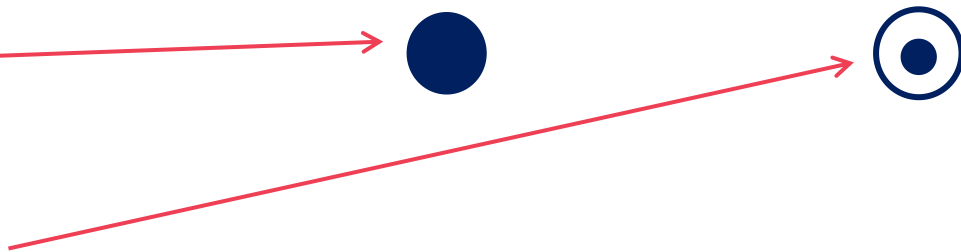
# Diagramme d'activités

## Initialisation et état de fin

❑ **État initial** : unique.



❑ **État final** : a priori unique



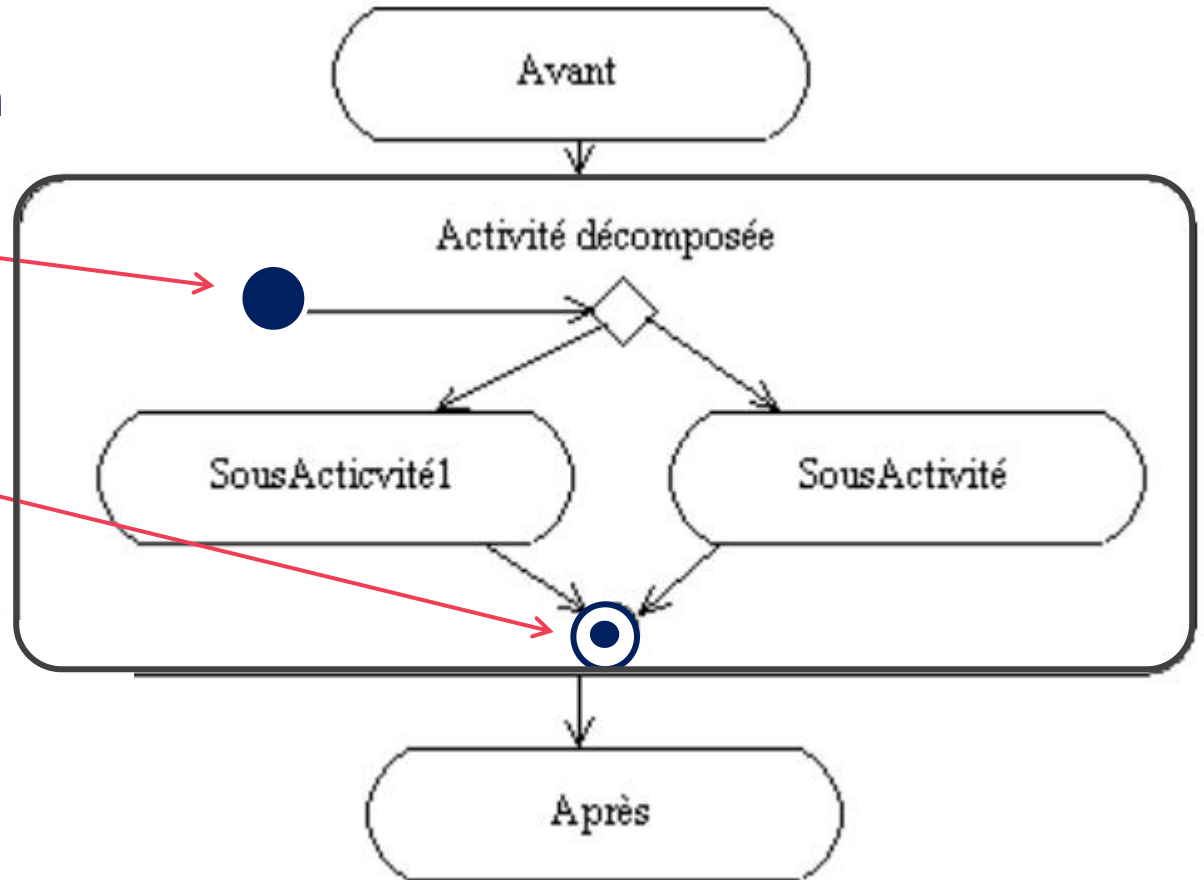
Plusieurs possibles si fin dans des états différents, pas forcément d'état final (processus infini).

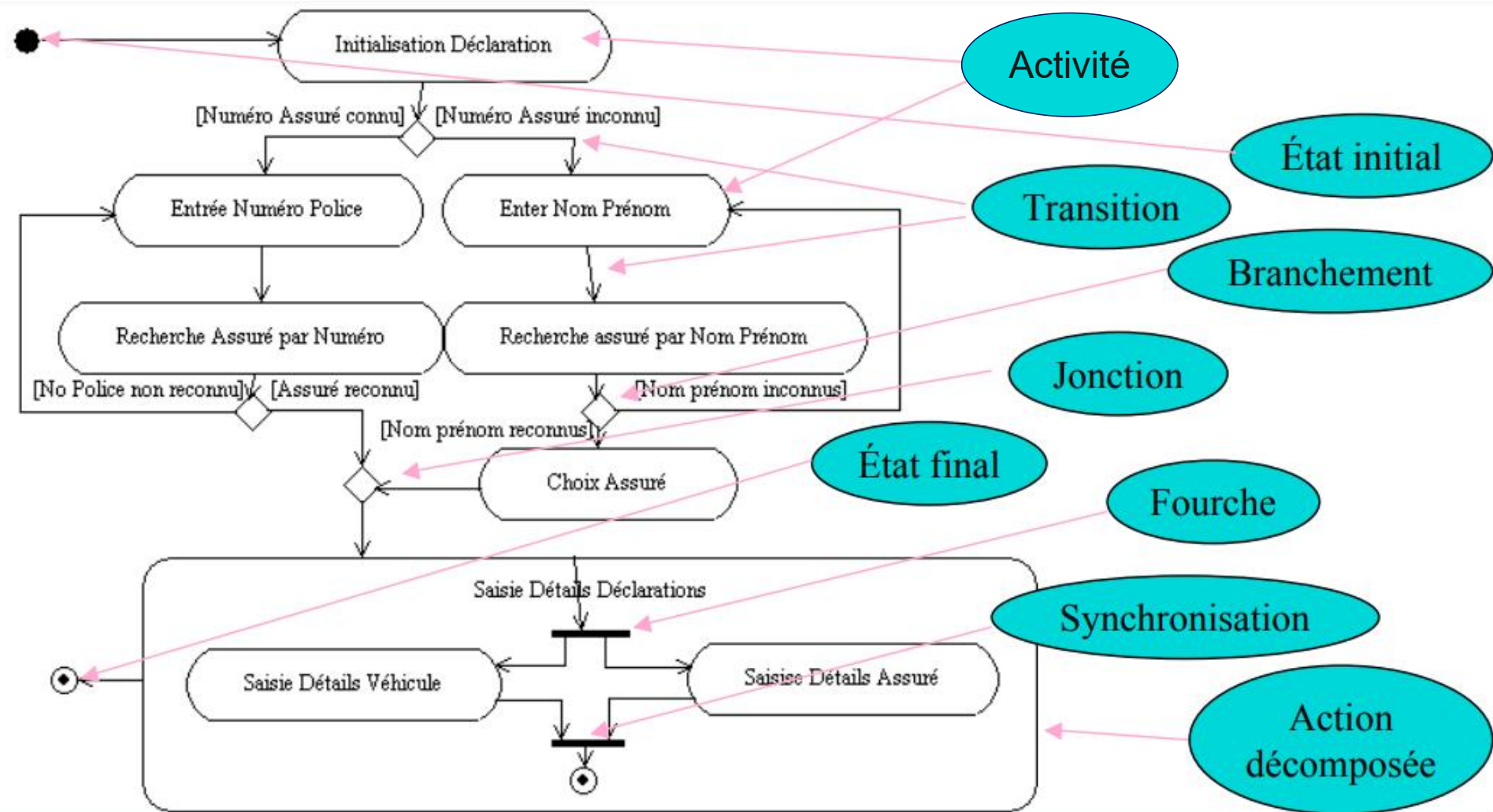
# Diagramme d'activités

## Initialisation et état de fin

❑ État initial

❑ État final





# Diagramme d'activités

## Composants supplémentaires

 **Action**

 **Couloirs de poste**

 **Flux d'objet**

# Diagramme d'activités

## Composants supplémentaires

☐ **Action**

☐ Couloirs de poste

☐ Flux d'objet



# Diagramme d'activités

## Action

- ❑ **Activité** : Une activité représente l'exécution d'un traitement non atomique.
- ❑ **Action** : Une action représente l'exécution d'un traitement atomique.
  - ❑ Cette exécution se traduit par un changement d'état du système ou le retour d'une valeur.
  - ❑ Les actions correspondent à l'appel d'une opération, l'envoi d'un signal, la création ou la destruction d'un objet ou encore l'évaluation d'une expression.

# Diagramme d'activités

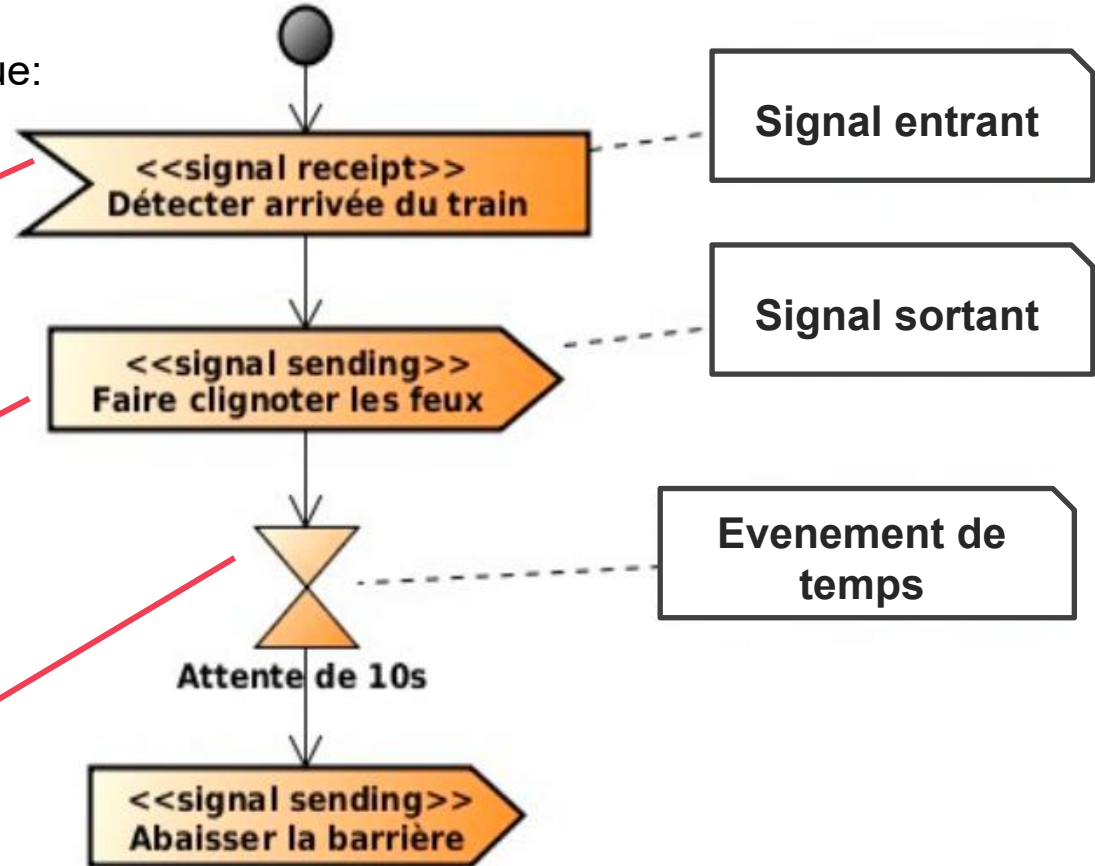
## Action

❑ Exemple et représentation graphique:

❑ Réception par une activité d'un signal extérieur.

❑ Emission d'un signal vers l'extérieur par une activité.

❑ Action automatique qui concerne le temps.



# Diagramme d'activités

## Composants supplémentaires

 **Action**

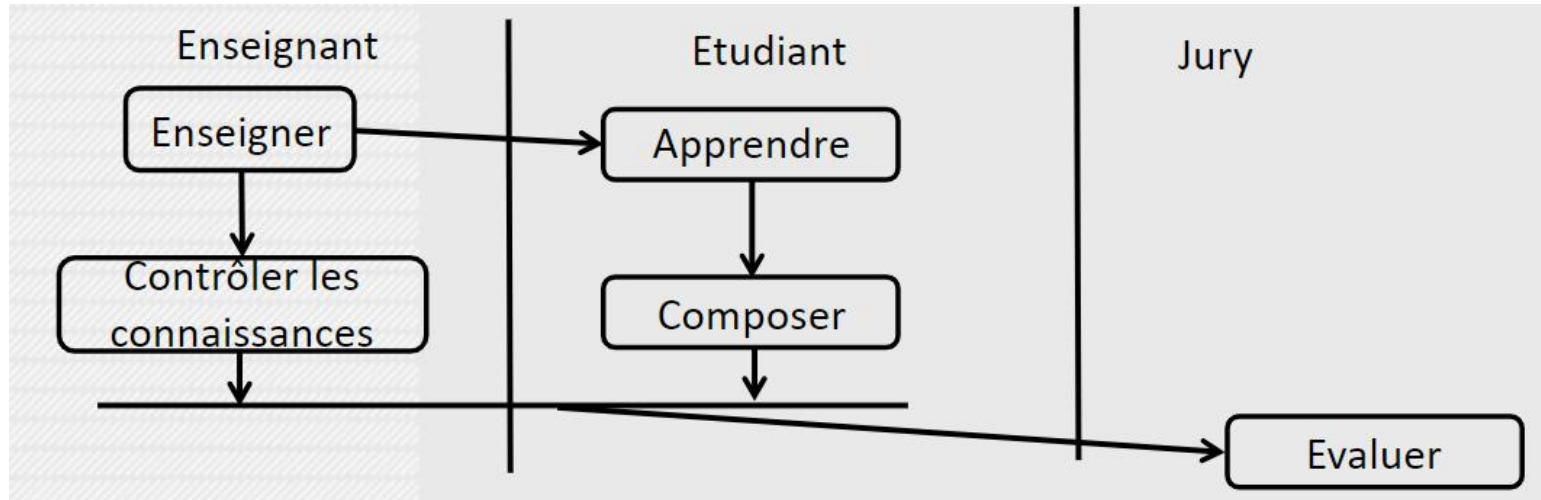
 **Couloirs de poste**

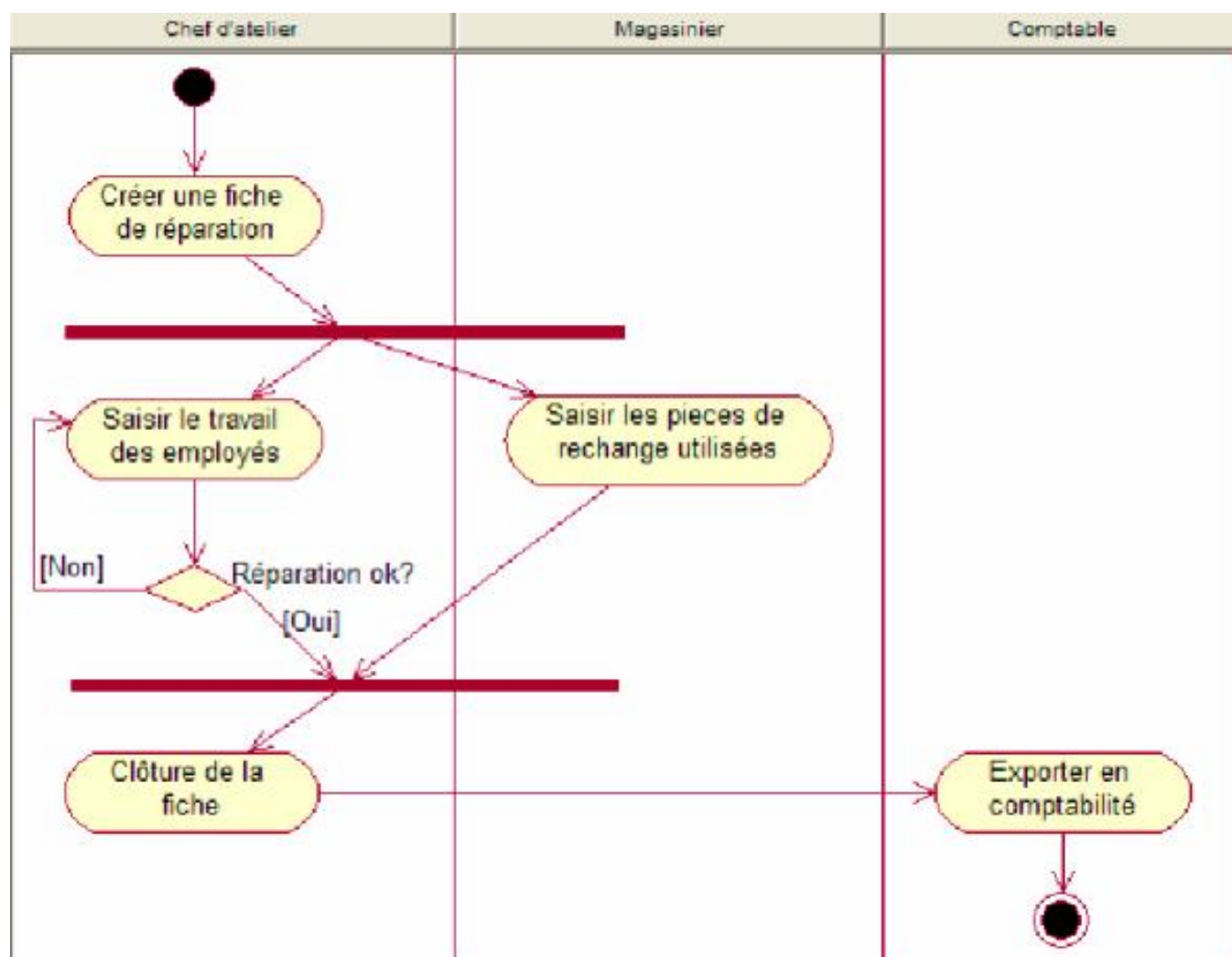
 **Flux d'objet**

# Diagramme d'activités

## Couloirs de poste

- ❑ Explicitation de l'agent effectuant les activités
- ❑ Chaque activité est allouée à un couloir correspondant à la ressource concernée : Partenaire, Tavailleur, Enseignant.





# Diagramme d'activités

## Composants supplémentaires

 Action

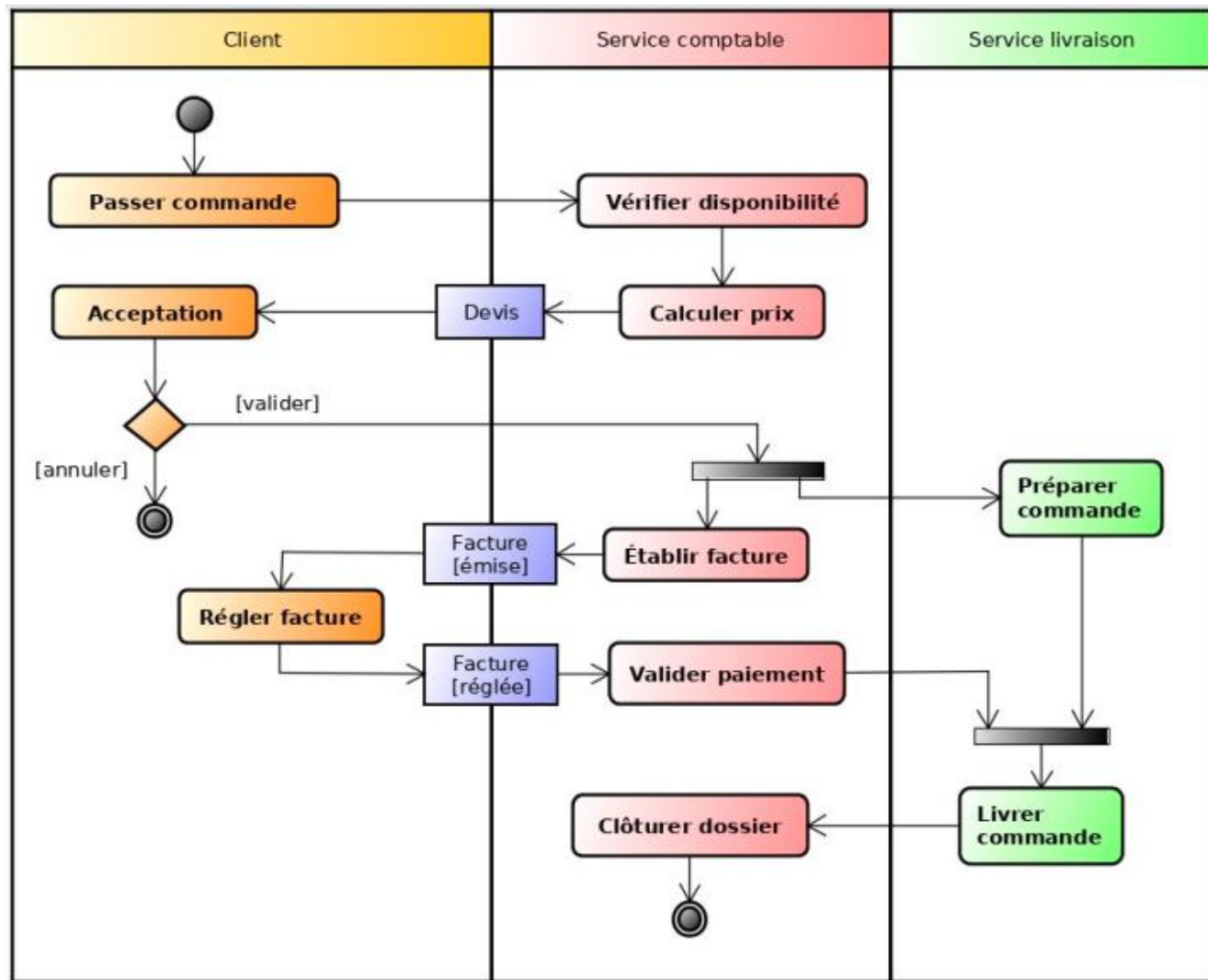
 Couloirs de poste

 **Flux d'objet**

# Diagramme d'activités

## Flux d'objet

- ❑ Transmission d'un objet entre deux activités sur une transition.
- ❑ L'état [état] de l'objet peut être précisé.





# Exercice

Construire un diagramme d'activité pour modéliser le processus de commander d'un produit.

Le processus concerne les acteurs suivants:

- ❑ Client: qui commande un produit et qui paie la facture
- ❑ Caisse: qui encaisse l'argent du client
- ❑ Vente: qui s'occupe de traiter et de facturer la commande du client
- ❑ Entrepôt: qui est responsable de sortir les articles et d'expédier la commande.

# Exercice

Créer un diagramme d'activité pour la gestion « **Créer une fiche de réparation** »

Pour créer une fiche de réparation, le chef d'atelier saisit les critères de recherche de voitures dans le système.

Le logiciel de gestion des réparations lui donne la liste des voitures correspondant aux critères entrés. Si la voiture existe, le chef d'atelier va sélectionner la voiture. Le logiciel va, ensuite, fournir les informations sur le véhicule. Si la voiture est sous garantie, le chef devra saisir la date de demande de réparation. Si la voiture n'existe pas, le chef va saisir les informations concernant ce nouveau véhicule. Dans tous les cas, le chef d'atelier devra saisir la date de réception et de restitution. Si le dommage de la voiture est payé par l'assurance, le logiciel va fournir une liste d'assurances au chef d'atelier. Ce dernier sélectionnera l'assurance adéquate. Enfin, le logiciel enregistre la fiche de réparation. Si le dommage de la voiture n'est pas payé, le chef d'atelier envoie une demande de paiement.