

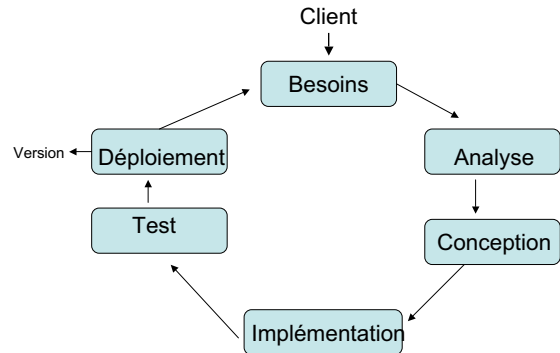
# Unified Modeling Language UML

Salima Hassas

Cours sur la base des transparents de : Gioavanna Di Marzo Serugendo et Frédéric Julliard

1

## Cycle de vie du logiciel



2

## Développement Logiciel

### • A faire

- Comprendre et conceptualiser le problème (besoins, analyse)
- Résoudre le problème (conception)
- Donner une solution (implémentation)
- Documenter

### • UML permet de

- Spécifier, visualiser et comprendre le problème
- Capturer, communiquer et utiliser des connaissances pour la résolution du problème
- Spécifier, visualiser et construire la solution
- Documenter la solution

3

## UML: définition

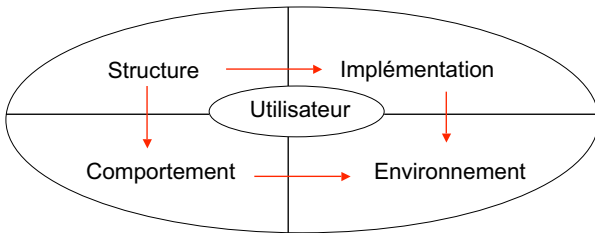
### • Unified Modeling Language

- Langage servant à décrire des modèles d'un système matériel ou logiciel basé sur des concepts orienté-objets
- Système de notations pour modéliser les systèmes en utilisant des concepts orienté-objets
- Langage de modélisation visuelle (graphique) qui permet de:
  - Spécifier le problème et sa solution
  - Visualiser le problème et la solution sous différents angles
  - Construire la solution
  - documenter

4

## Modèles et Vues

- Les modèles du système sont visualisés par des vues
- Chaque vue correspond à une facette différente du système
- Chaque participant au développement du système en a une vue différente



5

## Les vues d'UML (1/3)

- Vue Utilisateur
  - Buts et objectifs des clients du systèmes
  - Besoins requis par la solution
- Vue structurelle
  - Aspects statiques représentant la structure du problème
- Vue Comportementale
  - Aspects dynamiques du comportement du problème et de sa solution
  - Interactions et collaborations entre éléments de la solution

6

## Les vues d'UML (2/3)

- Vue implémentation
  - Aspects de la structure et du comportement de la solution
- Vue environnementale
  - Aspects de la structure et du comportement du domaine dans lequel est réalisée la solution

7

## Les vues d'UML (3/3)

- Les vues doivent être consistantes entre elles
- Elles doivent accompagner le cycle de vie du logiciel
- La modification d'une vue implique la modification des autres vues

8

## Les diagrammes d'UML

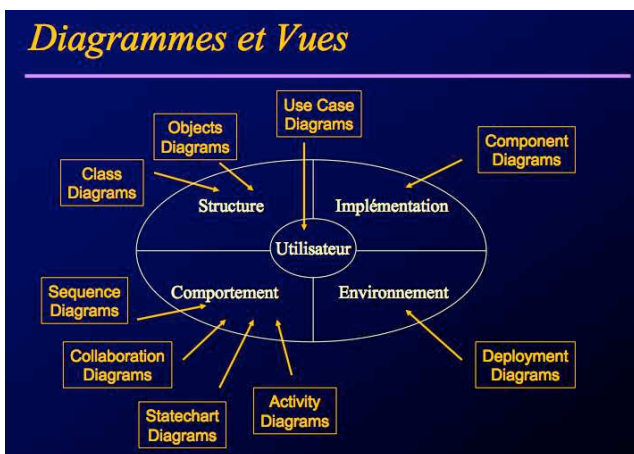
- Les diagrammes sont des graphes (nœud et arcs) qui permettent de représenter le problème et sa solution selon différents points de vue
- Les diagrammes forment ainsi des modèles du système (spécifier et visualiser)
- La combinaison de diagrammes représentent les vues du système
- Il existe 9 diagrammes représentant les deux aspects:
  - Statiques et structurels : relations
  - Dynamiques: comportement, collaborations, responsabilités

9

## Les diagrammes d'UML

- 5 diagrammes structurels (vue statique)
  - Cas d'utilisation (Use case)
  - Classes
  - Objets
  - Composants
  - Déploiement
- 4 diagrammes comportementaux (vue dynamique)
  - Séquence
  - Activités
  - États -Transitions (Stateschart)
  - Collaboration

10



11

Diagrammes de Cas  
d'Utilisation

Use Cases

12

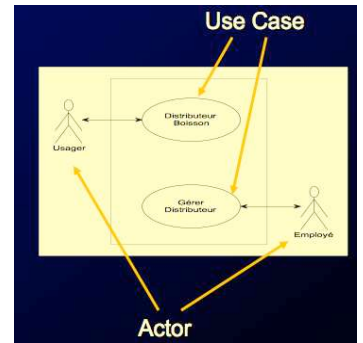
## Les Cas d'Utilisation

- Représenter les besoins
  - La phase d'analyse des besoins nécessite de comprendre les besoins, de les exprimer et les formaliser
- Moyens pour représenter les besoins en UML :
  - Diagramme de **cas d'utilisation**: exprime l'organisation de l'utilisation du système par ses acteurs
  - Diagramme de **séquence**: pour chaque cas d'utilisation donne une description temporelle de l'interaction d'un acteur avec le système (**scénario**)
  - Diagramme **Objets/classes**: informations échangées entre système et acteurs
  - Diagramme de **collaboration**: interactions entre les objets métiers du système

13

## Les Cas d'Utilisation (Use Case : Jacobson 92)

- Décrit la fonctionnalité que le système délivre à ses utilisateurs (humains ou autre système) et les liens qui peuvent exister entre eux (include, uses ou extends)
- Acteur (Actor) : Rôle joué par un utilisateur du système
- Cas d'utilisation (Use case): séquence d'actions que le logiciel garantit. Il définit les besoins du client



14

## Les Cas d'Utilisation (Use Case : Jacobson 92)

### • Acteur: définition

- Entité externe (humain ou système) qui agit selon un rôle sur le système
- Identifié par un nom correspondant à son rôle
- Il peut être accompagné d'une description textuelle du rôle

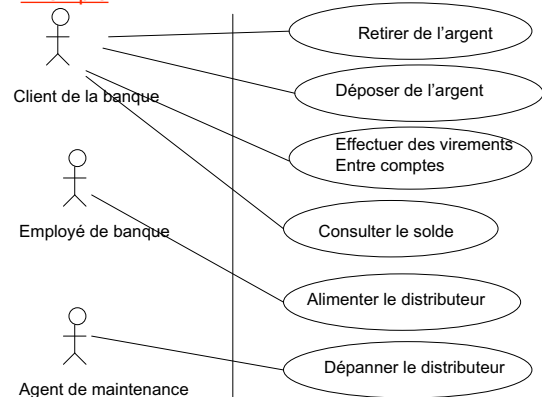
### • Cas d'utilisation: définition

- ensemble des actions réalisées par le système en réponse à une action d'un acteur
- Suite d'interactions entre un acteur et le système
- Correspond à une fonction visible par l'utilisateur
- Permet d'atteindre un objectif aux yeux de l'utilisateur
- Doit être utile
- Les cas d'utilisation ne doivent pas se chevaucher

15

## Les Cas d'Utilisation (Use Case : Jacobson 92)

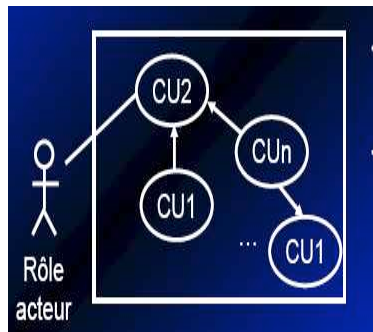
### Exemple



16

## Les Cas d'Utilisation (Use Case : Jacobson 92)

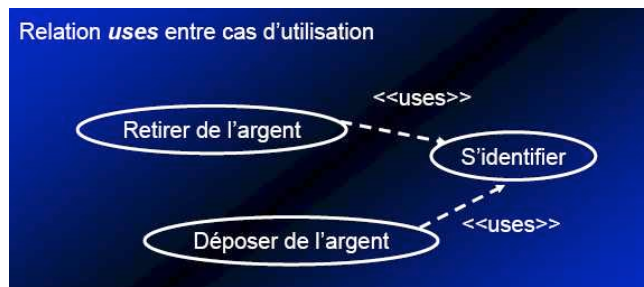
- Relations entre cas d'utilisations
  - Il n'existe pas de communications entre les cas d'utilisation d'un système mais simplement des relations d'utilisation (uses ou include) ou d'extension (extends)
  - Les communications entre acteurs ne sont pas représentées



17

## Les Cas d'Utilisation (Use Case : Jacobson 92)

- Relations entre cas d'utilisations



18

## Les Cas d'Utilisation (Use Case : Jacobson 92)

- Relations entre cas d'utilisations

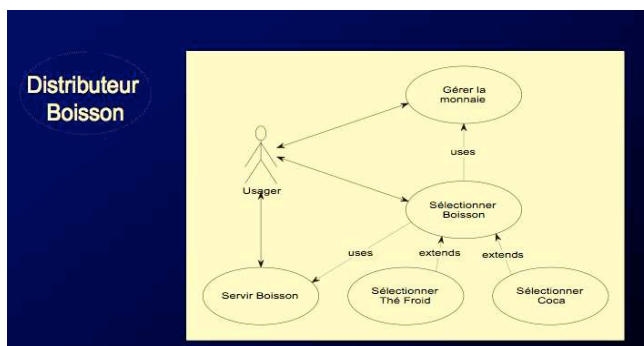
### Relation *extends* entre cas d'utilisation



19

## Les Cas d'Utilisation (Use Case : Jacobson 92)

- Relations entre cas d'utilisations



20

## Les Cas d'Utilisation (Use Case : Jacobson 92)

### • Cas d'utilisation et scénario

- le **système** = **ensemble** de **cas d'utilisation**
- le système possède les **cas d'utilisation** mais pas les **acteurs**
- Un **cas d'utilisation** = ensemble de « chemins d'exécution » possibles
- Un **scénario** = un chemin particulier d'exécution  
= une **séquence d'événements**
- Un **scénario** = Instance de cas d'utilisation
- Une **instance d'acteur** créer un **scénario**

21

## Les Cas d'Utilisation (Use Case : Jacobson 92)

### • Cas d'utilisation et scénario

un **scénario** peut être représenté par diagramme de **séquence** qui décrit un échange particulier entre un ou plusieurs acteurs et le système :

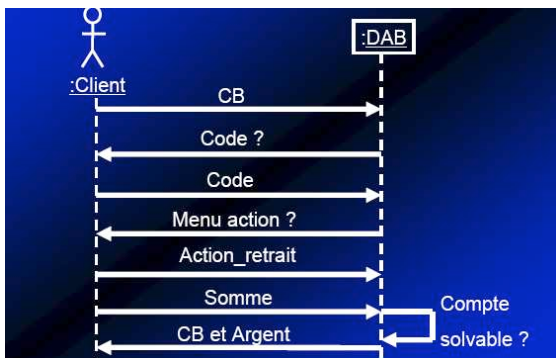
- nature des infos **échangées** entre des **instances** d'acteurs ou d'objets du système
- aspect **temporel** : **flot ordonné d'événements**

un **scénario** peut également être représenté par un diagramme de **collaboration** (cf. **Chapitre IV**)

22

## Les Cas d'Utilisation (Use Case : Jacobson 92)

### • Cas d'utilisation et scénario





23

Diagramme de Classes  
et  
diagramme d'objets

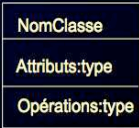
24

## Diagramme de Classes

- ♦ Décrivent la **structure statique** du système à l'aide de classes, packages, et relations
- ♦ **Nœuds:**
  - Packages
 

NomPackage
  - Interfaces
 

NomInterface

NomMethod
  - Classes: Attributs, Opérations (visibilité et paramètres)
 

NomClasse

Attributs:type




Opérations:type

Classes: Attributs, Opérations (visibilité et paramètres)

- + public
- - privé
- # protégé
- package

25

## Diagramme de Classes

- ♦ **Arcs:**
  - dépendance
 
    - référence à des classes ou objets passés en paramètres ou statiques (« use »)
    - relations entre package
  - association
 
    - navigabilité entre objets, message entre objets
    - flèche est optionnelle
  - agrégation
 
    - « has-a »

26

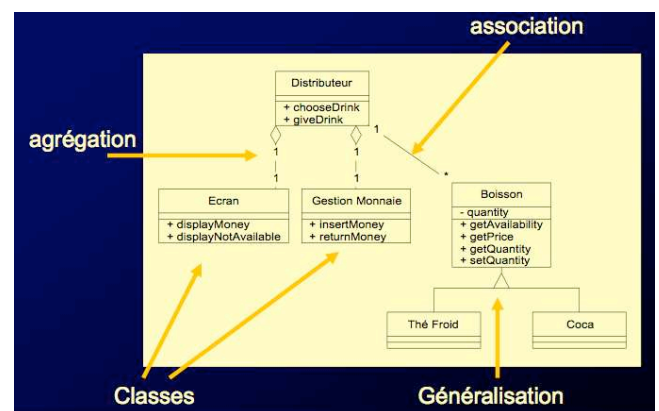
## Diagramme de Classes

- composition
 
  - « has-a » avec responsabilité sur durée de vie
- généralisation
 
  - héritage
- réalisation
 
  - implements
  - réalisation d'un use case

- ♦ Les liens contiennent la multiplicité d'objets associés

27

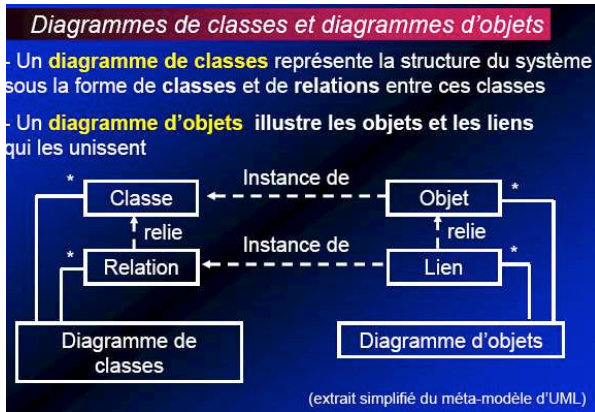
## Diagramme de Classes



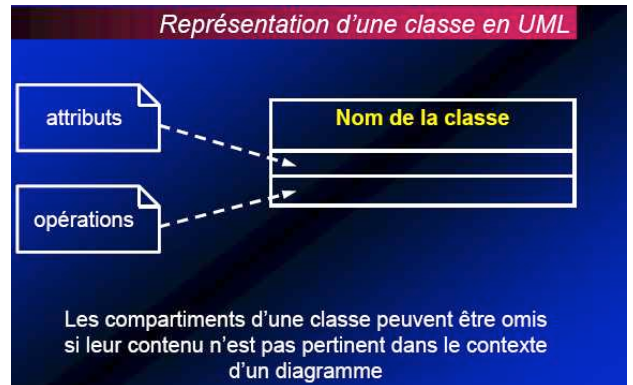
28



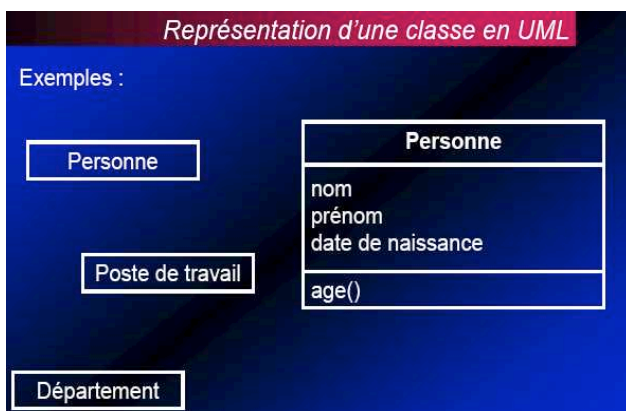
### Diagramme de Classes



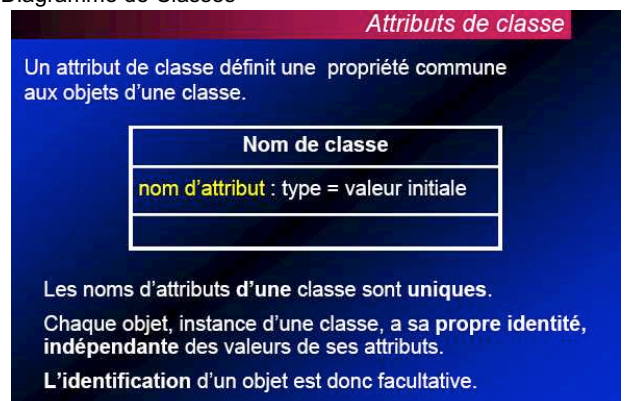
### Diagramme de Classes



### Diagramme de Classes

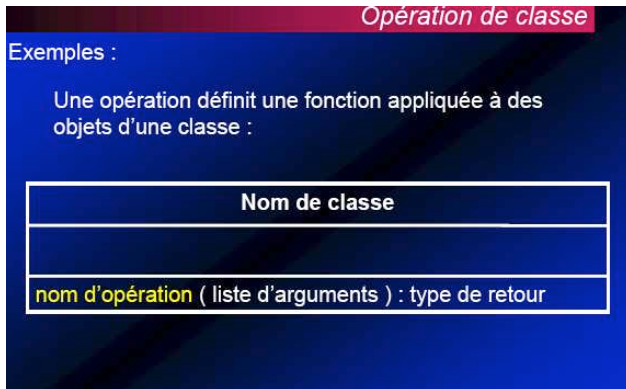


### Diagramme de Classes



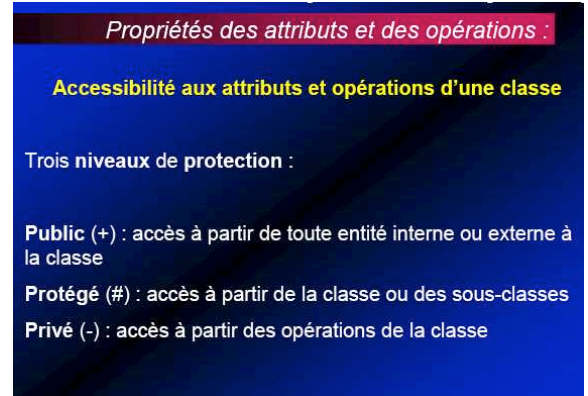


## Diagramme de Classes



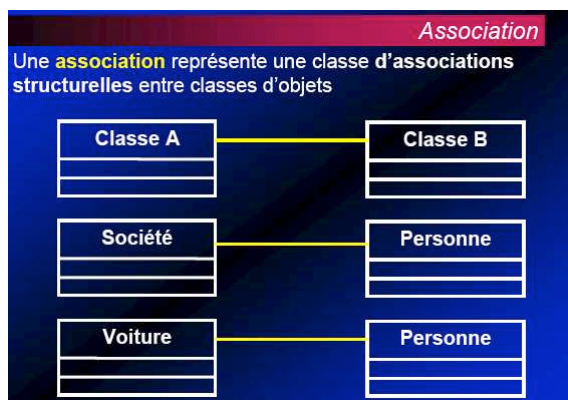
33

## Diagramme de Classes



34

## Diagramme de Classes



35

## Diagramme de Classes



36

## Diagramme de Classes

*Nommage des rôles*

- Toute association binaire possède 2 rôles
- un rôle définit la manière dont une classe intervient dans une relation
- Le nommage des associations et le nommage des rôles ne sont pas exclusifs l'un de l'autre

```

classDiagram
    class Société
    class Personne
    Société --> Personne : Travaille pour
    Société -- employeur
    Personne -- employé
    
```

- Intérêt des rôles dans le cas où plusieurs associations lient deux classes : distinction des concepts attachés aux associations

37

## Diagramme de Classes

*Association réflexive*

```

classDiagram
    class Personne
    Personne --> Personne : Enfants
    Personne -- Parents : 2
    
```

Nommage des rôles indispensable à la clarté du diagramme

38

## Diagramme de Classes

*Multiplicité des associations*

La **multiplicité** est une **information portée par le rôle**, qui quantifie le **nombre de fois** où un **objet** participe à une **instance de relation**

1	un et un seul
0 .. 1	zéro ou un
M .. N	de M à N (entiers naturels)
*	de zéro à plusieurs
0 .. *	de zéro à plusieurs
1 .. *	de un à plusieurs
N	exactement N (entier naturel)

39

## Diagramme de Classes

*Multiplicité des associations*

```

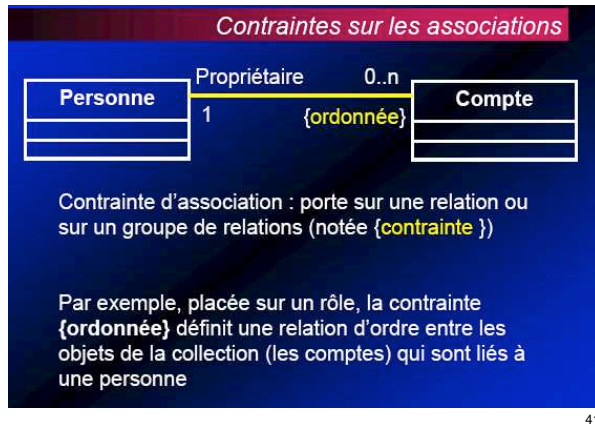
classDiagram
    class Personne
    class Société
    Personne -- Société : Employé
    Personne -- 0..* : Employé
    Société -- 1 : Employeur
    
```

1 : Chaque personne travaille pour une et une seule société (toute les personnes ont un emploi)

0 .. \* : Une société emploie de zéro à plusieurs personnes

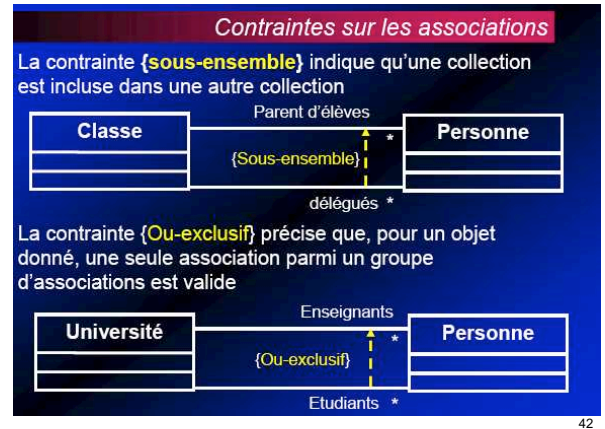
40

## Diagramme de Classes



41

## Diagramme de Classes



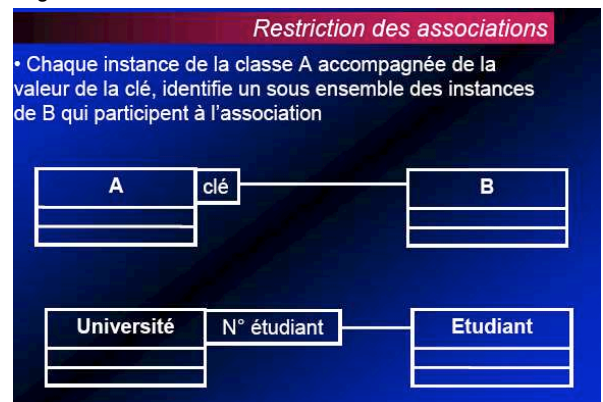
42

## Diagramme de Classes



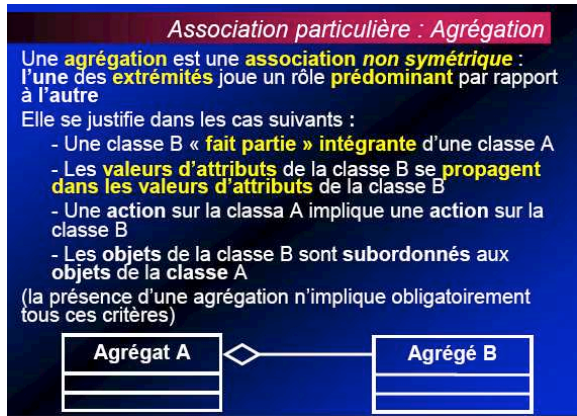
43

## Diagramme de Classes



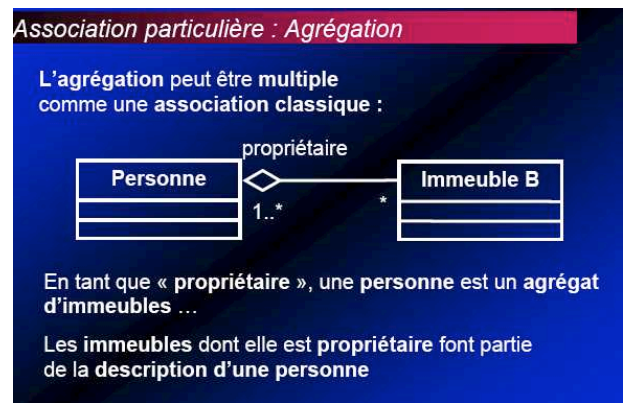
44

## Diagramme de Classes



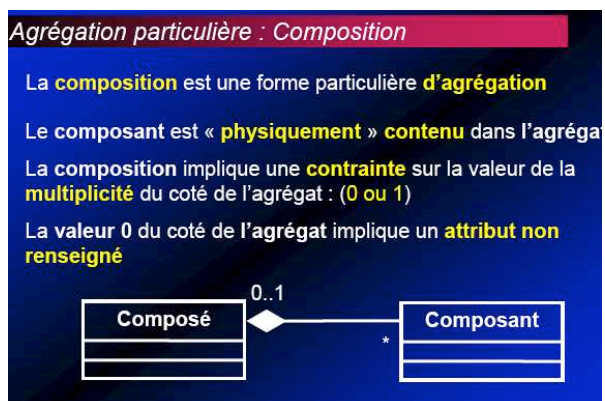
45

## Diagramme de Classes



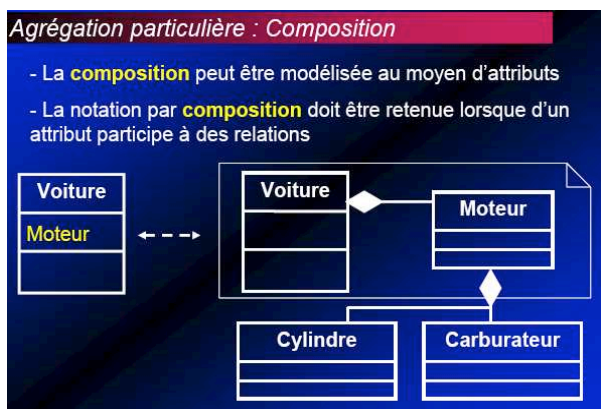
46

## Diagramme de Classes



47

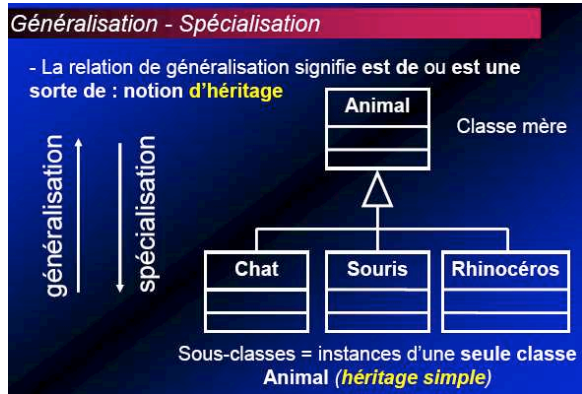
## Diagramme de Classes



48

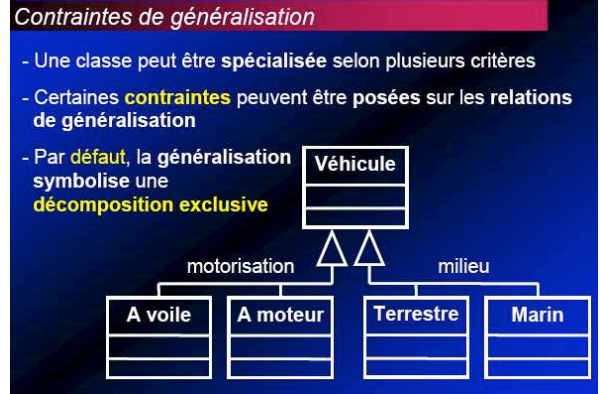


## Diagramme de Classes



49

## Diagramme de Classes



50

## Diagramme d'objets

- ♦ Exemples de class diagrams avec des instances d'objets

- ♦ **Nœuds:** objets

:Class	Object	Object:Class
Attributs:valeur	Attributs:valeur	Attributs:valeur

- ♦ **Arcs:** relations entre objets, instances d'associations
- ♦ Respecter les multiplicités d'objets associés aux arcs (dans le class diagram)

51

## Diagramme d'objets

- Représente les liens **structurél** entre instances de classes
- Facilite la **compréhension** de **structures complexes**
- Trois **représentations** possibles des instances :

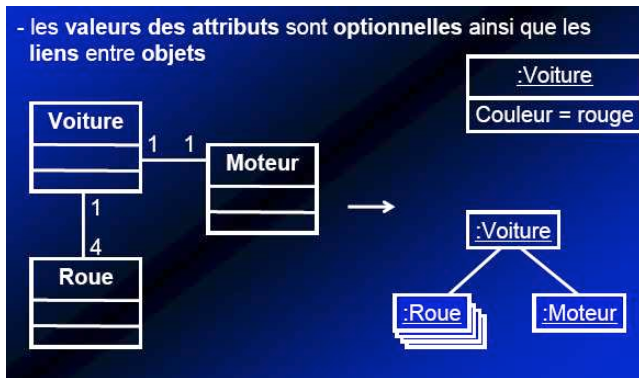
Nom de l'objet

Nom de l'objet:NomClasse

:NomClasse

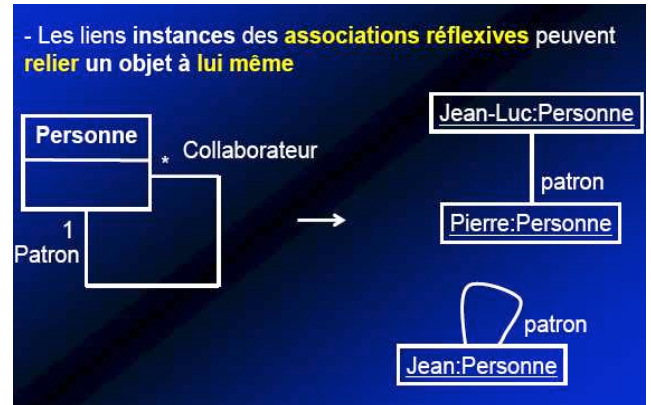
52

Diagramme d'objets



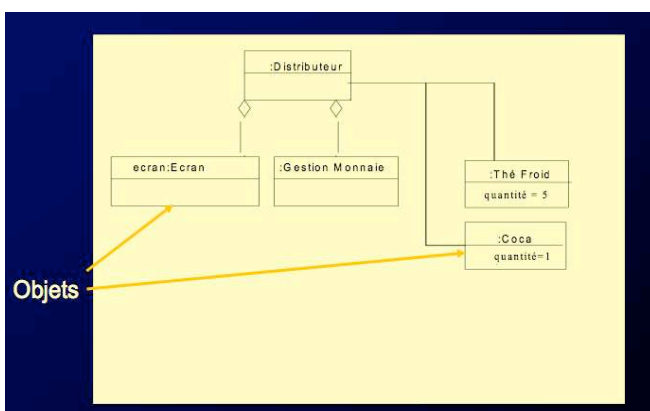
53

Diagramme d'objets



54

Diagramme d'objets



55

## Diagramme de séquence

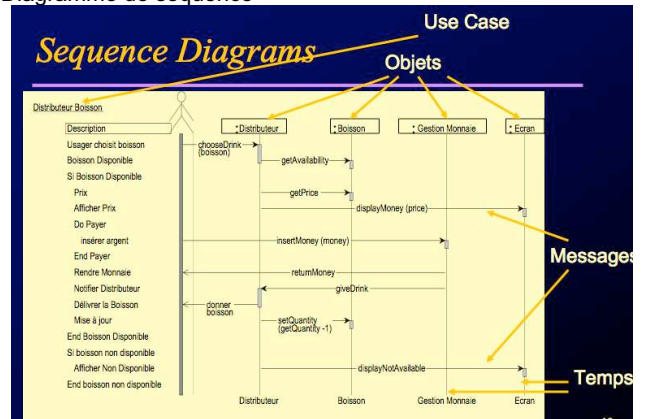
56

## Diagramme de séquence

- ♦ Etablissent le lien entre *Use Case* et *Class Diagrams*
  - ♦ Décrivent l'échange de messages entre classes
  - ♦ **Class rôles**
    - objets participant à l'interaction
- :Class    Object    Object:Class
- ♦ **Lignes de temps**
  - ♦ **Messages** activant des opérations chez l'objet receveur
    - représente la communication entre objets
  - ♦ **Scénario**: cas particulier de séquence

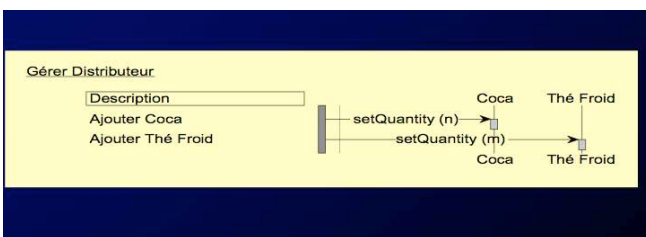
57

## Diagramme de séquence



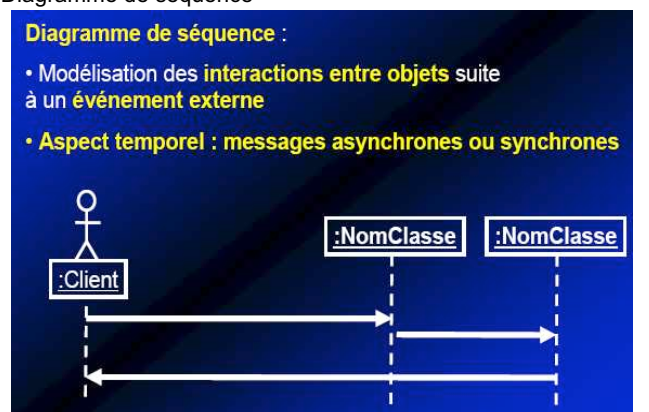
58

## Diagramme de séquence



59

## Diagramme de séquence



60



## Diagramme de séquence

### Catégories de messages

#### 2 catégories de messages :

- **synchrone** : l'émetteur est bloqué jusqu'au traitement effectif du message
- **asynchrone** : l'émetteur n'est pas bloqué, il peut poursuivre son exécution

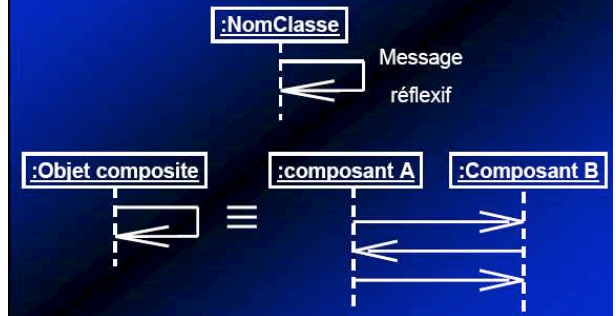


61

## Diagramme de séquence

### Envoi de messages d'un objet sur lui-même

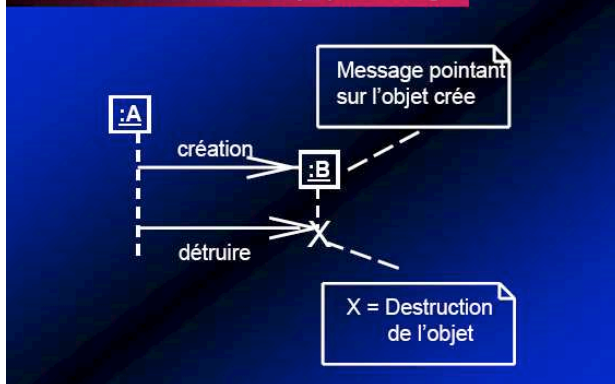
Un objet peut s'envoyer des messages à lui-même :



62

## Diagramme de séquence

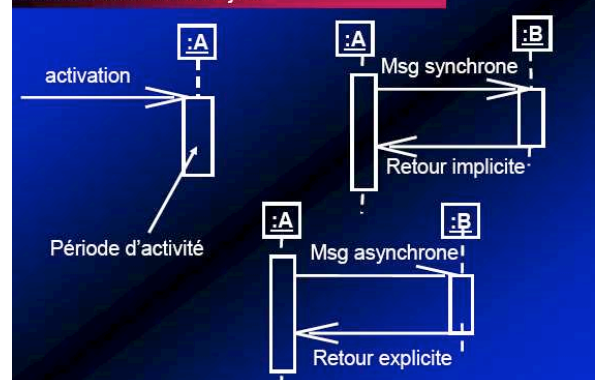
### Création et destruction d'un objet par message



63

## Diagramme de séquence

### Période d'activité des objets



64

Diagramme de séquence

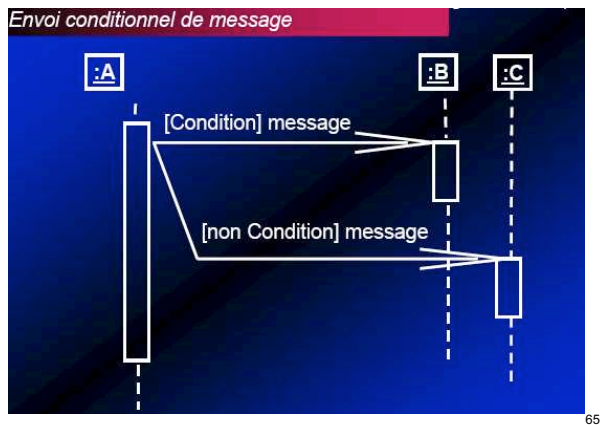


Diagramme de séquence

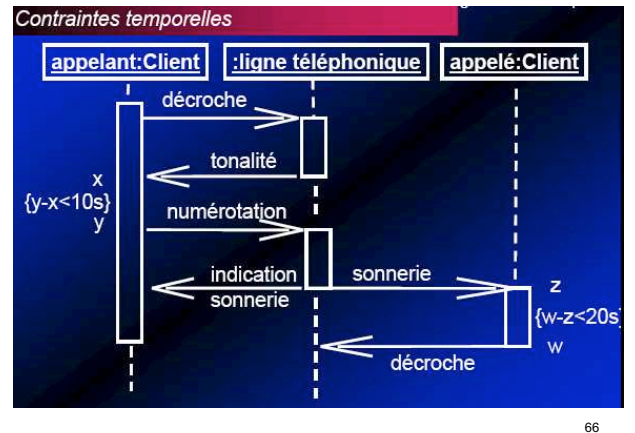
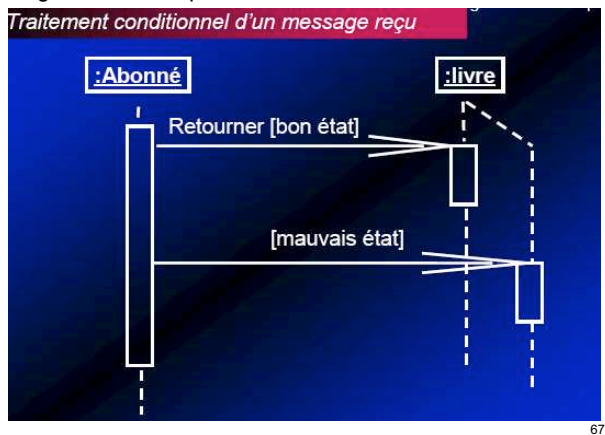


Diagramme de séquence



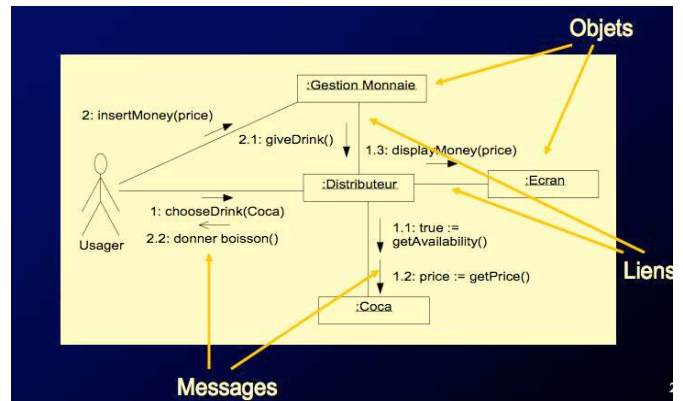
## Diagramme de Collaboration

## Diagramme de Collaboration

- ◆ Décrivent les échanges de *messages* entre classes, et définissent les associations
  - sémantiquement équivalents aux sequence diagrams, mais ...
    - sequence diagrams illustrent l'ordre des événements
    - collaboration diagrams représentent les interconnexions entre objets et sont visuellement différents
- ◆ **Class roles**: objets participant à l'interaction
- ◆ **Liens**: instances d'associations
- ◆ **Messages**: envoyés le long des liens
- ◆ **Scénarios**: cas particulier

69

## Diagramme de Collaboration



70

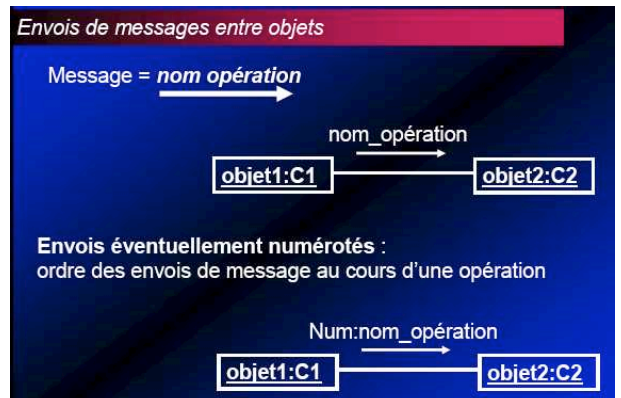
## Diagramme de Collaboration

Diagramme de collaboration (d'objets) : extension des diagrammes d'objets : vue *dynamique*

- Décrit le **comportement collectif** d'un **ensemble d'objets**,
- en vue de **réaliser une opération**
- en décrivant leurs **interactions** modélisées par des **envois** (éventuellement **numérotés**) de **messages**

71

## Diagramme de Collaboration



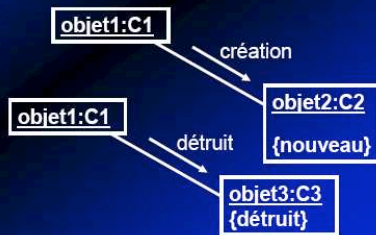
72

## Diagramme de Collaboration

### Contraintes associées aux envois de message

Les objets (et les liens) **créés** ou **détruits** au cours d'une interaction peuvent **respectivement** porter les contraintes :

- {Nouveau}
- {Détruit}



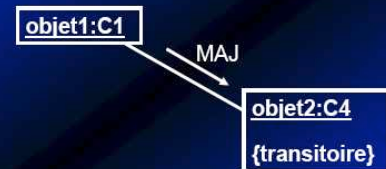
73

## Diagramme de Collaboration

### Contraintes associées aux envois de message

Les objets (et les liens) **créés** et **détruits** au cours de la même interaction porte la contraintes :

- {transitoire}

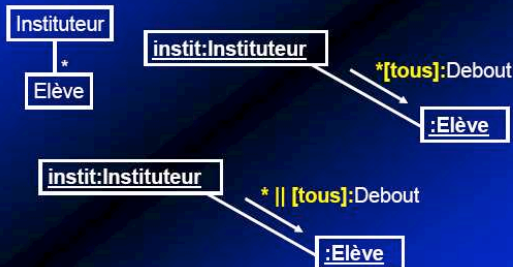


74

## Diagramme de Collaboration

### Itérations dans un diagramme de collaboration

Possibilité d'exprimer l'envoi répétitifs de messages (éventuellement en parallèle) sur une collection d'objets

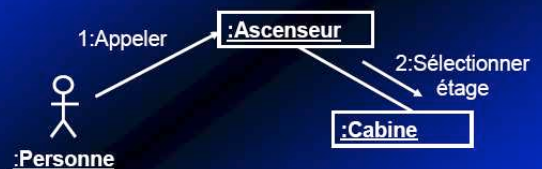


75

## Diagramme de Collaboration

### Itérations dans un diagramme de collaboration

Il est possible de faire intervenir un **acteur** (cf. chapitre III) dans un diagramme de collaboration : afin de représenter le **comportement** du système sous l'effet d'un **stimuli externe**



76



## Diagramme de Collaboration

### Itérations dans un diagramme de collaboration

Les **objets** qui contrôlent le flot sont dits **actifs**

Un **objet actif** peut **activer** un **objet passif** en lui envoyant un **message**. Une fois le message **traité**, le flot de **contrôle** est **restitué** à l'**objet appelant**

**Ex : photocopieuse**

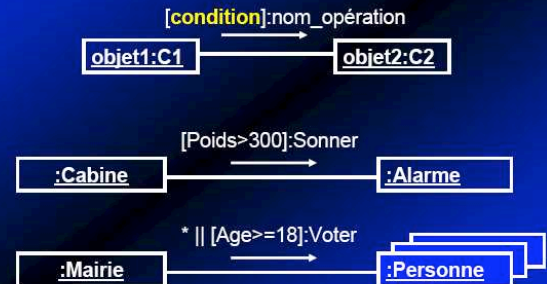


77

## Diagramme de Collaboration

### Conditions sur les envois de message

L'envoi d'un message peut être assorti d'une condition

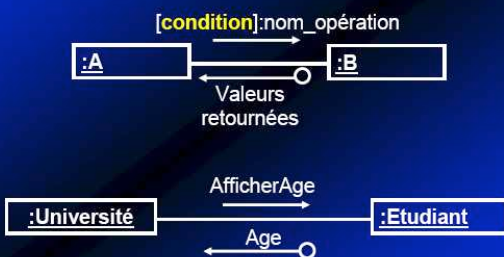


78

## Diagramme de Collaboration

### Retour d'une liste de valeurs à l'issue d'un envoi de message

Une liste de valeurs peut être retournée suite à l'envoi d'un message



79

## Diagramme de Collaboration

### Exemple : distributeur de boisson

**Diagramme de collaboration « demande d'une boisson disponible (café) avec introduction de la somme exacte »**



80

## Diagramme d'Etats-Transitions Statecharts

81

### Diagramme Etats-Transitions

- ♦ Décrivent les **états** et le **comportement** d'une *classe* en réponse à des événements *extérieurs*
- ♦ **Etats**
  - initial, final, intermédiaire
  - sous-états
- ♦ **Transitions**
  - nom d'événement / action

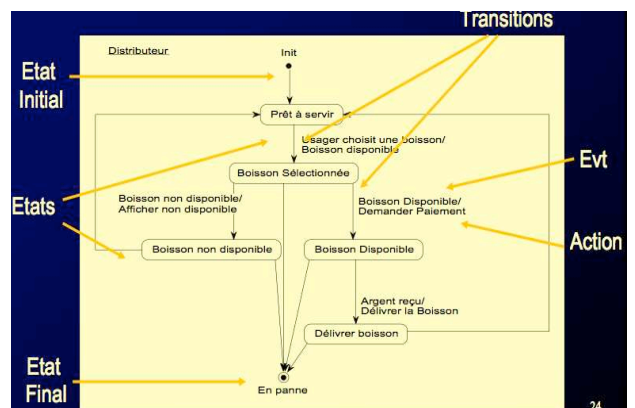
82

### Diagramme Etats-Transitions

- décrit le **comportement** des objets d'une classe au moyen d'un automate d'états associé à la classe
- Le comportement est modélisé par un graphe :
  - **Nœuds** = états possibles des objets
  - **Arcs** = transitions d'état à état.
- Une transition :
  - = **exécution d'une action**
  - = **réaction de l'objet sous l'effet d'une occurrence d'evt**

83

### Diagramme Etats-Transitions



84

## Diagramme Etats-Transitions

### Notion d'état

- un **état** = **étape** dans le cycle de vie d'un objet durant lequel
  - il **satisfait à certaines conditions**
  - il **réalise certaines actions**
  - ou **attend certains événements**
- chaque **objets** possède à un **instant donné** un **état particulier**
- chaque **état** est **identifié par un nom**
- un **état** est **stable et durable**

85

## Diagramme Etats-Transitions

### Notion d'état

- Chaque diagramme d'états-transitions comprend **un état initial**.
- Pour un **niveau hiérarchique donné**, il y a **un et un seul état initial**, mais **plusieurs états finaux** correspondant chacun à une fin de vie de l'objet différente.
- Il est possible de n'avoir **aucun état final** : ex : un système que ne s'arrête jamais.



86

## Diagramme Etats-Transitions

### Notion d'événement

- un **événement** correspond à l'**occurrence d'une situation donnée** dans le domaine étudié
- un **événement** est une **information instantanée** qui doit être **traitée dans l'instant où il se produit**
- l'**événement** est **déclencheur de la transition d'état à état**. Un objet, placé dans un **état donné**, attend l'**occurrence d'un événement** pour passer dans un autre état



87

## Diagramme Etats-Transitions

### Notion d'événement

- **syntaxe d'un événement** :
- Nom de l'événement (Nom de paramètre : Type, ...)
- La description complète d'un évt est donnée par :
- nom de l'événement
  - liste des paramètres
  - objet expéditeur
  - objet destinataire
  - sa description textuelle

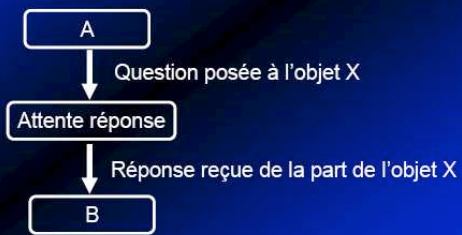
88



## Diagramme Etats-Transitions

### Communication entre objets par événements

- L'objet émetteur de la requête se met en attente de la réponse de l'objet récepteur de la requête



89

## Diagramme Etats-Transitions

### Notion de garde

- Une **garde** est une **condition booléenne** qui permet ou non le déclenchement d'une transition lors de l'occurrence d'un événement



90

## Diagramme Etats-Transitions

### Communication entre objets par événements

- Les **gardes** permettent de **conserver la propriété de déterminisme** d'un automate d'états finis.
- Lorsqu'un occurrence d'événement survient, les **gardes**, qui doivent être **mutuellement exclusives**, sont évaluées.
- Le **résultat** de cette évaluation permet de **valider** puis de **déclencher** une transition possible



91

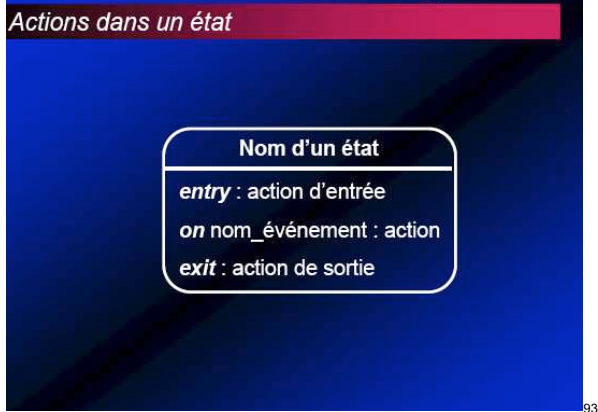
## Diagramme Etats-Transitions

### Actions dans un état

- Les états peuvent également contenir des actions :
  - elles sont exécutées
    - à l'entrée ou à la sortie de l'état ou
      - l'action d'**entrée (entry)** est exécutée de manière **instantanée et atomique**
      - l'action de **sortie (exit)** est exécutée à la sortie de l'état
    - lorsqu'une occurrence d'événement interne survient
      - l'action sur un événement interne (**on**) est exécutée lors de l'occurrence d'un événement qui ne conduit pas à un autre état

92

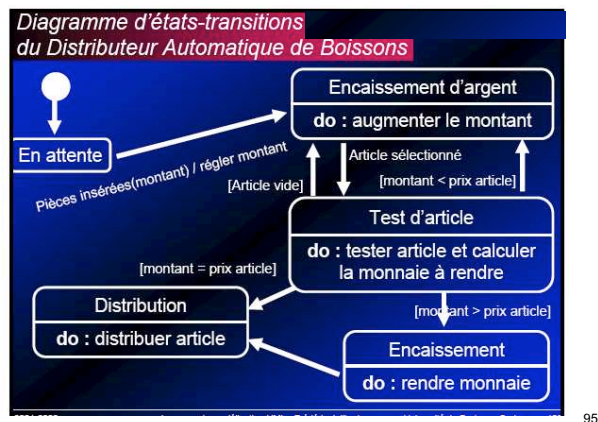
## Diagramme Etats-Transitions



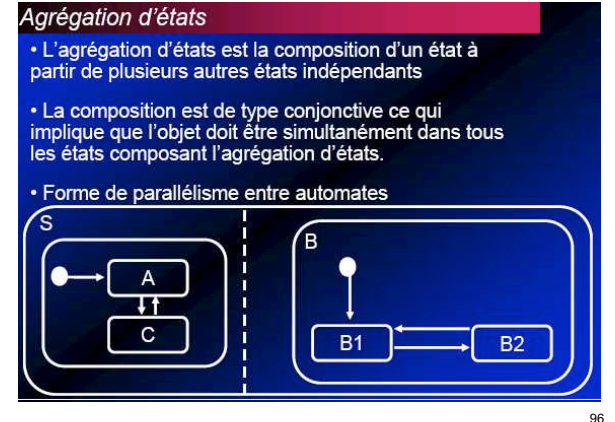
## Diagramme Etats-Transitions



## Diagramme Etats-Transitions



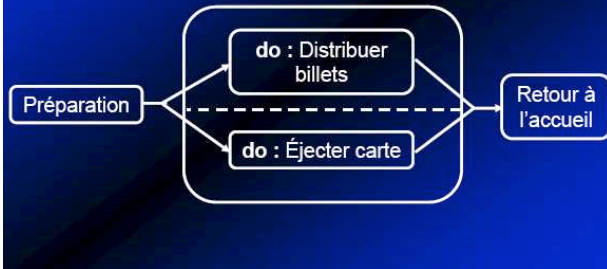
## Diagramme Etats-Transitions



## Diagramme Etats-Transitions

### Agrégation d'états

- Exemple : activité d'émission de billets



97

## Diagramme d'activités

98

## Diagramme d'activités

- ◆ Décrivent le *comportement* d'une *classe* en réponse à des calculs *internes*
- ◆ Similaire aux statecharts, mais pour les événements internes (et non extérieurs)

99

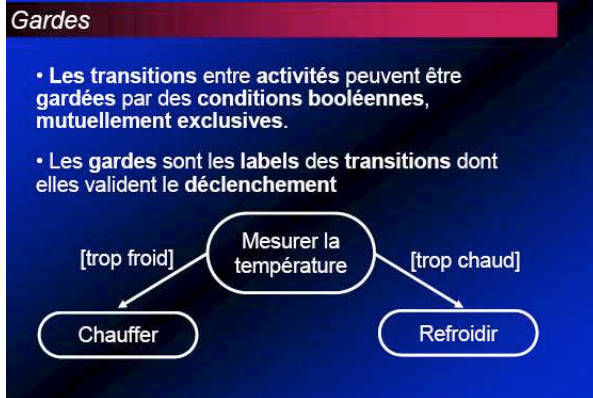
## Diagramme d'activités

- Variante des diagrammes d'états-transitions : ce diagramme met l'accent sur les activités, leurs relations et leurs impacts sur les objets



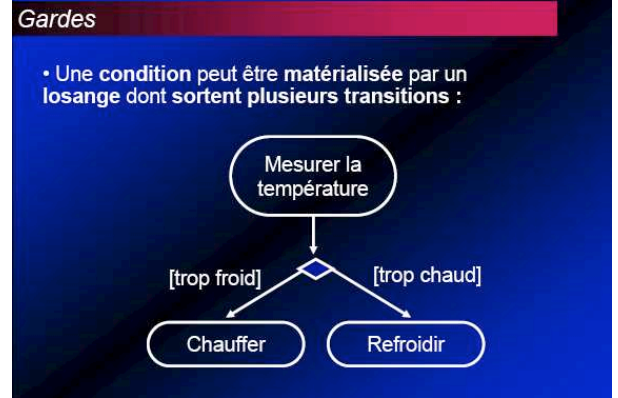
100

Diagramme d'activités



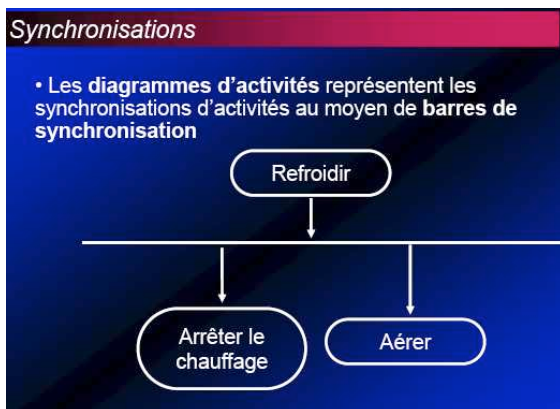
101

Diagramme d'activités



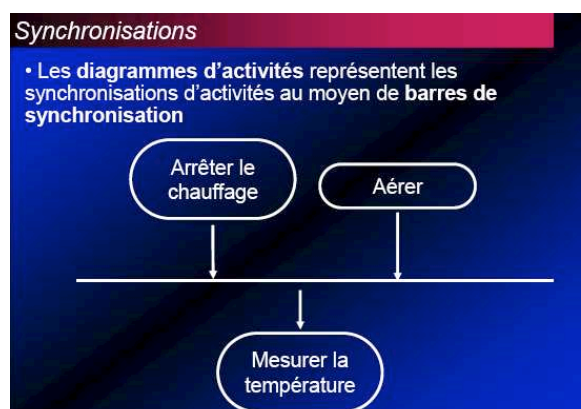
102

Diagramme d'activités



103

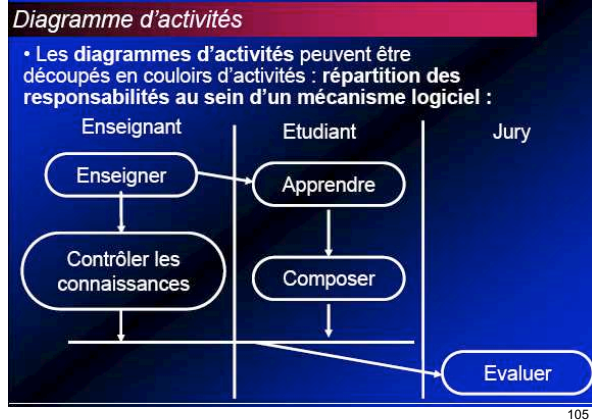
Diagramme d'activités



104

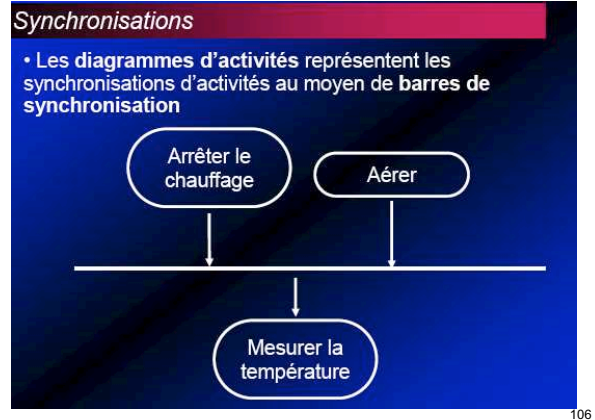


## Diagramme d'activités



105

## Diagramme d'activités



106

## Autres diagrammes

- ♦ **Component Diagrams**
  - Décrivent l'organisation des composants et les dépendances qui les lient
- ♦ **Deployment Diagrams**
  - Décrivent les ressources de calcul, leurs configurations et le lien entre les exécutable et les ressources de calcul
- ♦ **OCL: Object Constraint Language**
  - Langage permettant d'exprimer des conditions attachées à des éléments d'un modèle
- ♦ **Stéréotypes**
  - nouveaux éléments peuvent être définis et introduits dans un modèle

107

## UML et méthodologies

- ♦ UML est une notation
- ♦ UML n'est pas une méthodologie
- ♦ Les méthodologies peuvent utiliser UML comme notation
  - RUP (Rational Unified Process)
  - OOSE (Object-Oriented Software Engineering, Jacobson)
  - OMT (Object Modeling Technique, Rumbaugh)

108