



LOG210

PLAN DES SÉANCES

Les liens pour les vidéos dans le format PDF de ce document sont sur le “[m]” des informations sur la durée de chaque vidéo. ex 22.5[m]

[Séance 1](#) | [Séance 2](#) | [Séance 3](#) | [Séance 4](#) | [Séance 5](#) |
[Séance 6](#) | [Séance 7](#) | [Séance 8](#) | [Séance 9](#) | [Séance 10](#) |
[Séance 11](#) | [Séance 12](#) | [Séance 13](#) |

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) |
[Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) |
[Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

1.1



SÉANCE 1

- Intro au cours
- Outils utilisés
- Analyse et conception orienté objet

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 2

• Modèle du domaine

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 3

- Diagramme de séquence système
- DSS et interface usagé

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 4

• RDCU sans les patron GRASP

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 5

• RDCU avec les patron GRASP

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 6

- PU
- Contrat

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 7

- Présentation invité
- HRC et le travail d'équipe
- Architecture en couche

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 8

- Présentation invité
- Test Driven Development (TDD)
- Dette technique

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) |
[DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) |
[Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) |
[TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) |
[Diagramme composant et déploiement](#) | [Diagrammes](#) |
Dette technique | Divers

SÉANCE

- FURPS
- Adaptateur, proxy...

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 10

- Relation entre gof et GRASP
- ? DDD

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 11

• Diagramme d'état

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 12

• Diagramme d'activité

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

SÉANCE 13

- Diagramme de composant et déploiement
- Diagrammes
- Divers

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

MODULE INTRODUCTION LOG210

ANALYSE ET CONCEPTION DE LOGICIELS

1. Présentation de l'équipe
2. Présentation personnelle 7.48m
3. Présentations des étudiants
4. Faire un logiciel pour un domaine que vous ne connaissez pas

Il faut trouver un moyen pour faire un lien entre ces titres de section et la section correspondante dans les notes de cours.

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

2.1

ÉQUIPE



- Responsable du cour
 - Christopher Fuhrman
 - christopher.fuhrman@etsmtl.ca



2.2

- Chargé de cours

- Yvan Ross, M.ing
- cc-yvan.ross@etsmtl.ca



Utiliser Discord pour communiquer avec moi.

2.3

COMMUNIQUER AVEC MOI



- Par discord: objet précis, ex. **GR01: question sur le modèle du domaine**
- Rencontres organisées de façon proactive en **dehors** de ces heures. Proposez moi au moins deux périodes de disponibilité.
 - Lun 8h30-12:00
 - Lun 13:30 - 17:00
 - Mar 18:00 - 21:30
 - Jeu 08:30 - 12:00
 - Ven 13:30 - 16:30

CHARGÉS DE LABORATOIRE



- 01 Jeu 08:30 - 11:30 Labo A-3322
 - hind.errahmouni.1@ens.etsmtl.ca
- 02 Lun 13:30 - 16:30 Labo A-3322
 - edouard.laforgue.1@ens.etsmtl.ca
- 03 Mer 18:00 - 21:00 Labo A-3322
 - islem.saidani.1@ens.etsmtl.ca

Utiliser Discord pour communiquer avec eux à propos du laboratoire.

PRÉSENTATION PERSONNELLE



2.6

CHRISTOPHER FUHRMAN



2.7

PRÉSENTATION PERSONNELLE

- Professeur depuis 2001 (Conception de logiciels)
- Expérience industrielle
 - 2019 : Développement Pharo.org Inria Lille
 - 2014-20 : PlantUML Gizmo
 - 2008 : Développement convertisseur XML FBI-WVU
 - 2000 : Directeur du programme NASA « ASSET »
 - 1998-1999 : Développeur chez ProLogic
 - 1989-1991 : Développeur chez Apple

PRÉSENTATION PERSONNELLE



- Diplômes universitaires
 - 1988 Bachelor of Science (informatique)
WVU
 - 1996 Docteur ès sciences (informatique technique) École Polytechnique Fédérale de Lausanne

YVAN ROSS



2 . 10

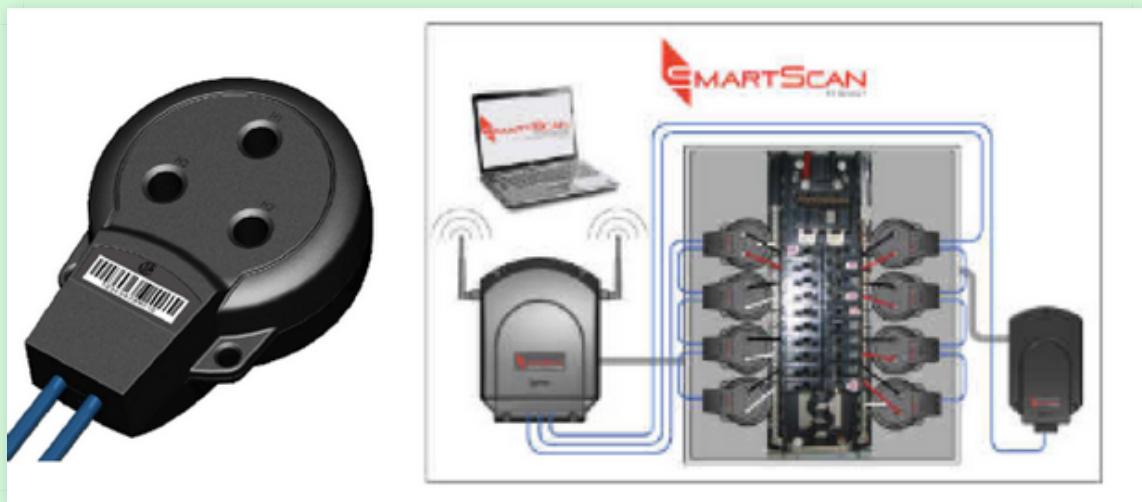
PRÉSENTATION PERSONNELLE



- Expérience industrielle
 - 2013-: Chargé de cours ETS
 - 2010-2015: Consultant
 - 2011-2013: Web analytique cloud (ROR), gestion de projet
 - 2009-2011: Reconnaissance vocale. C++, java, ROR, Perl

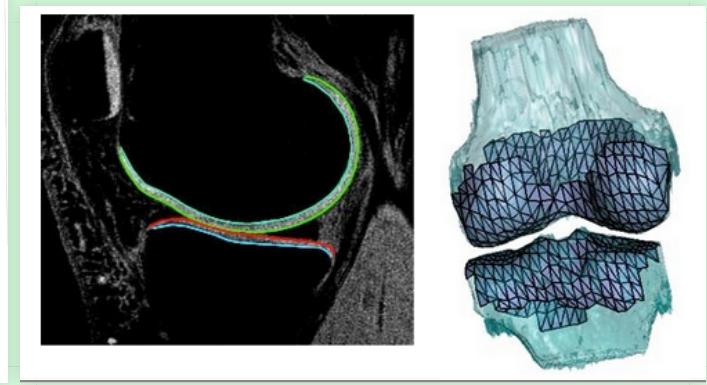
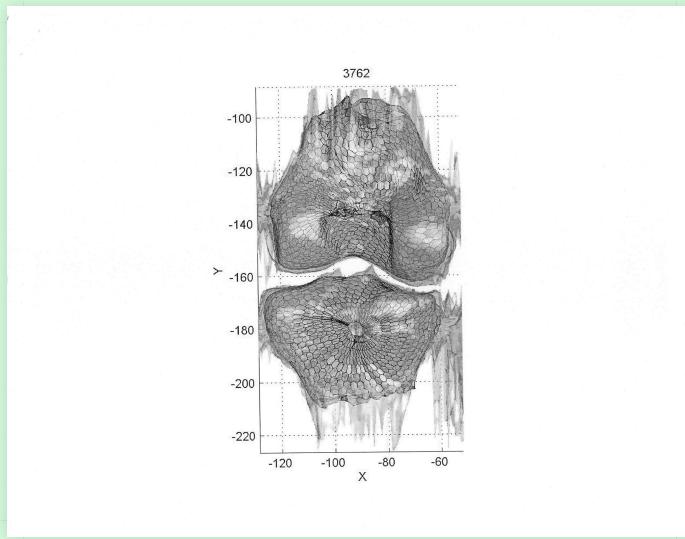
PRÉSENTATION PERSONNELLE (...)

- 2008-2009: Capteur d'analyse réseau électrique, C++, simulink , Matlab



PRÉSENTATION PERSONNELLE (...)

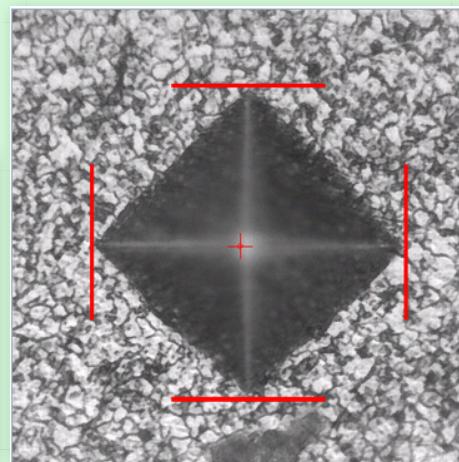
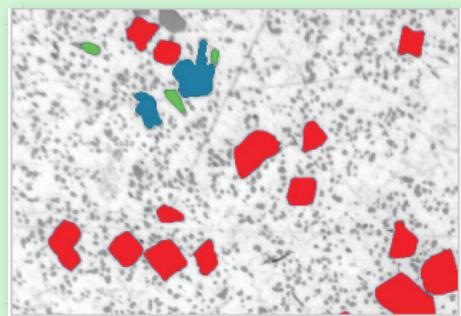
- 2001-2008: R&D en imagerie médicale, C++, Matlab



PRÉSENTATION PERSONNELLE (...)

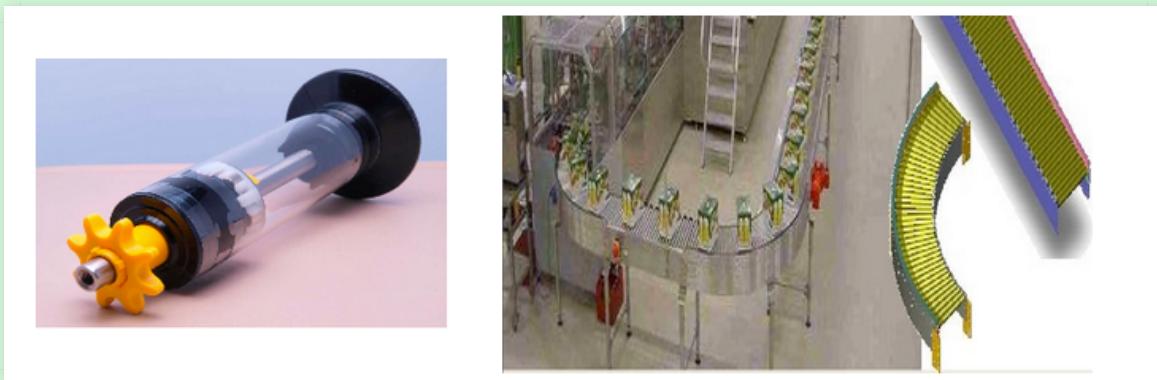


- 1998-2001: Développeur imagerie par microscopie, C++



PRÉSENTATION PERSONNELLE (..)

- 1994-1998: Directeur R&D, conception de système d'inspection automatisée par vision, C++, conception d'équipement de convoyage

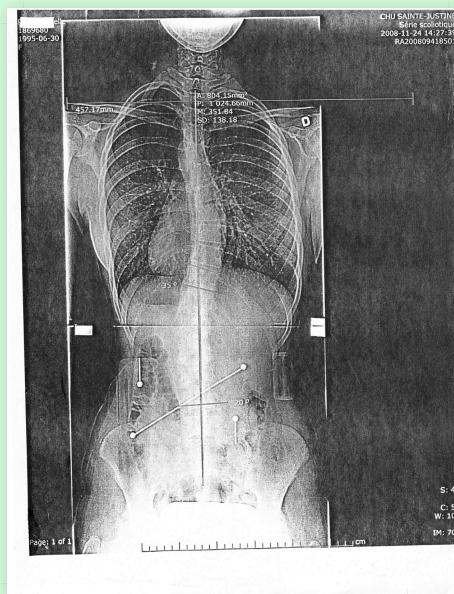


PRÉSENTATION PERSONNELLE (...)

DIPLOÔMES UNIVERSITAIRES



- 1994 Maîtrise en technologie des systèmes



DIPLÔMES UNIVERSITAIRES

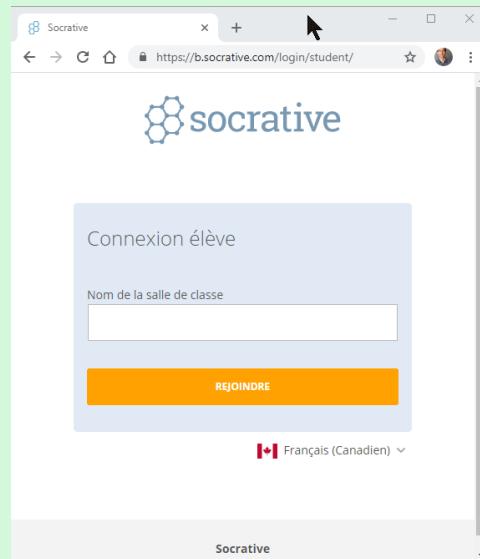
- 1990 Baccalauréat en Génie de la production automatisée



PRÉSENTATIONS DES ÉTUDIANTS

Questionnaire à tiny.cc/quizdesign

Nom de la salle de class: **ETSDESIGN**



2 . 18

PRÉSENTATIONS DES ÉTUDIANTS



- Comment développer un logiciel?
 - Quelle(s) méthode(s) de développement avez-vous suivie(s)?
 - Pour quel genre de logiciel?
- Quelles sont vos attentes pour LOG210?

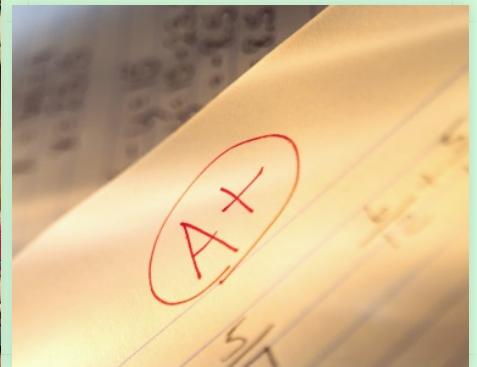
PRÉSENTATIONS DES ÉTUDIANTS

- Veuillez vous présenter sur Discord dans Général.
 - Votre nom, votre programme d'étude collégiale, votre programme d'étude universitaire, vos stages (entreprise, domaine, projet), pourquoi vous êtes en génie logiciel.



MEILLEURES PRATIQUES POUR APPRENDRE

Quelles sont les meilleures pratiques pour apprendre?



2.21

MES CONSEILS POUR RÉUSSIR



- L'étudiant qui réussit dans mes cours...

- ~~s'assied vers l'avant de la salle,~~
- est motivé pour apprendre et pour participer,
- est bien discipliné,
- est organisé,
- cherche de l'aide d'une façon proactive,
- sais que le multitâche nuit à la qualité.

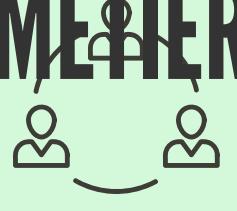
PROACTIF



1. **Domaine:** psychologie
2. **Auteur:** Office québécois de la langue française, 2002
3. **Définition:** Qui exerce un effet sur des faits ou des processus à venir

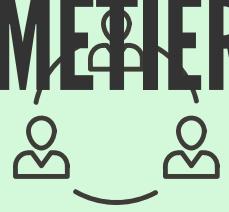
Exemple: Pompiers qui visitent les maisons pour vérifier les détecteurs de fumée

INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?



2 . 24

INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?



Revers de la médaille: certains reprochent à leur métier la nécessité de se mettre régulièrement au niveau. Ils trouvent difficile de suivre les évolutions avec les langages et les nouvelles pratiques qui ne cessent d'apparaître ou de se modifier



INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?



<http://www.developpez.com/actu/26910/Ingenieur-logiciel-le-meilleur-metier-au-monde-Oui-selon-une-etude-signee-CareerCast-com/>

2 . 25

INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?



Autre inconvénient: la solitude, notamment pour les développeurs qui préfèrent travailler depuis chez eux. Un choix qui leur offre un emploi du temps flexible, mais qui les mène à passer de longues heures sans interagir avec le moindre collègue.

<http://www.developpez.com/actu/26910/Ingenieur-logiciel-le-meilleur-metier-au-monde-Oui-selon-une-etude-signee-CareerCast-com/>

FAIRE UN LOGICIEL POUR UN DOMAINE QUE VOUS NE CONNAISSEZ PAS

Exemple:

Application Web permettant de planifier les formations (faites par des spécialistes en TI) des enseignants dans les écoles québécoises.

Comment procéder?

- Appliquer un processus pour la réalisation
 - Appliquer une méthodologie d'analyse
 - Appliquer une méthodologie de conception

MODULE OUTILS

1. Notes de cours
2. Outils de travail
3. Visual studio code
4. vscode: Markdown all in one
5. vscode: Plantuml
6. vscode: liveShare
7. plantuml website
8. Mastering markdown
9. Git: Fork and pull request 13.57m
10. Outils UML - 14.44m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

OUTILS DE TRAVAIL



- Plan de cours
- Moodle - classe inversé
- Examen - enquiz
- Google Drive pour le contenu
- Socrative
- Zoom / obs / youtube/ discord
- one note
- laboratoires

3.2

OUTILS DE TRAVAIL...



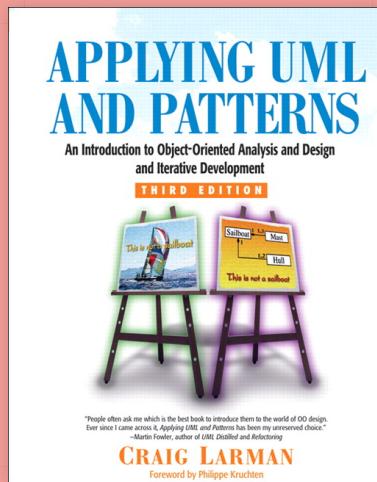
- Visual studio code
 - extension
 - <https://visualstudio.microsoft.com/services/share/>
 - plantuml
 - Markdown all in One
 - Markdown PDF
- <https://www.npmjs.com/package/gitinspector>
- <https://www.npmjs.com/package/tplant>
- <https://plantuml.com/>
- <https://wakatime.com/dashboard>

3.3

LIVRE OBLIGATOIRE



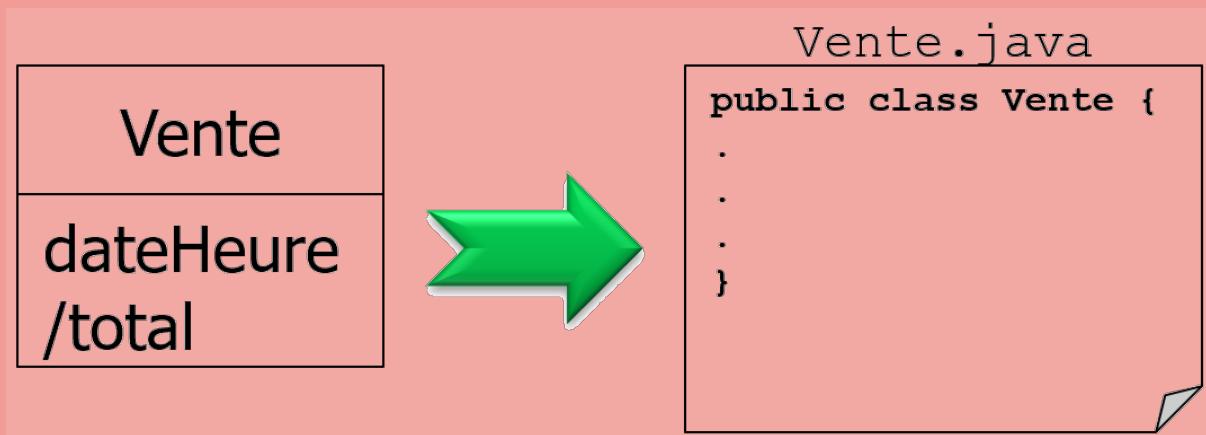
- Version anglaise ou française?



DÉFINITIONS



- Pro-ingénierie
 - Générer du code source à partir des diagrammes (UML)

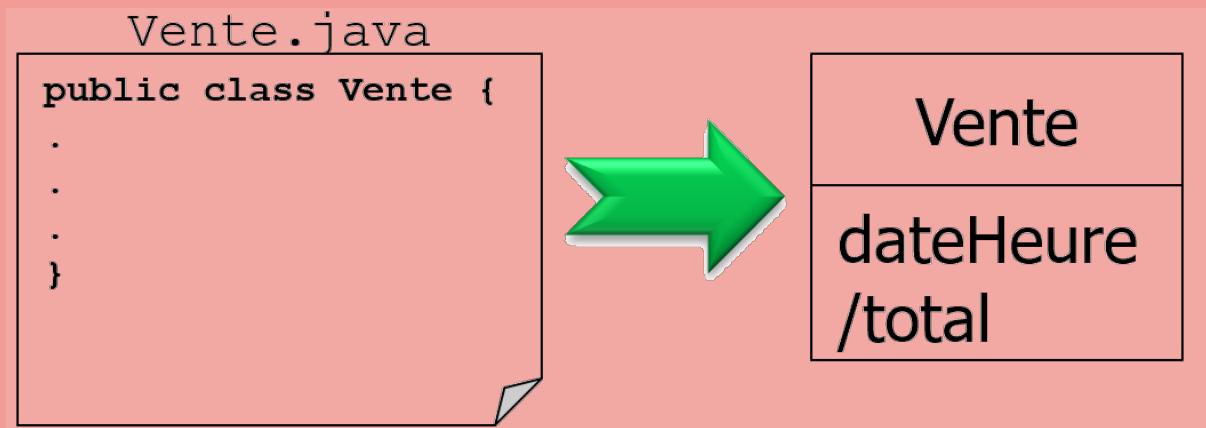


3.5

DÉFINITIONS



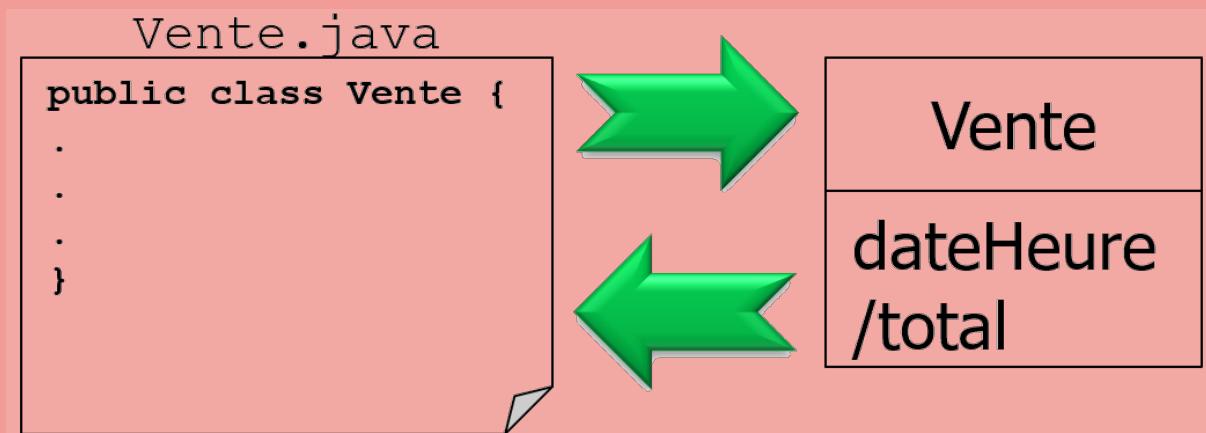
- Rétro-ingénierie
 - Générer des diagrammes (UML) à partir du code



DÉFINITIONS



- Ingénierie cycle
 - Fermer cette boucle



QUE DOIT-ON RECHERCHER DANS UN OUTIL ?



- Essayez un outil gratuitement avant de l'acheter.
- Testez l'outil sur un projet réel.
- Choisissez un outil qui s'intègre à votre IDE préféré (VS, Eclipse, etc.)
- Rétro-ingénierie est important
- L'impression sur grande feuilles est importante (Larman)

3 . 8

LISTE D'OUTILS UML



1. Wikipedia

- Suggestions:

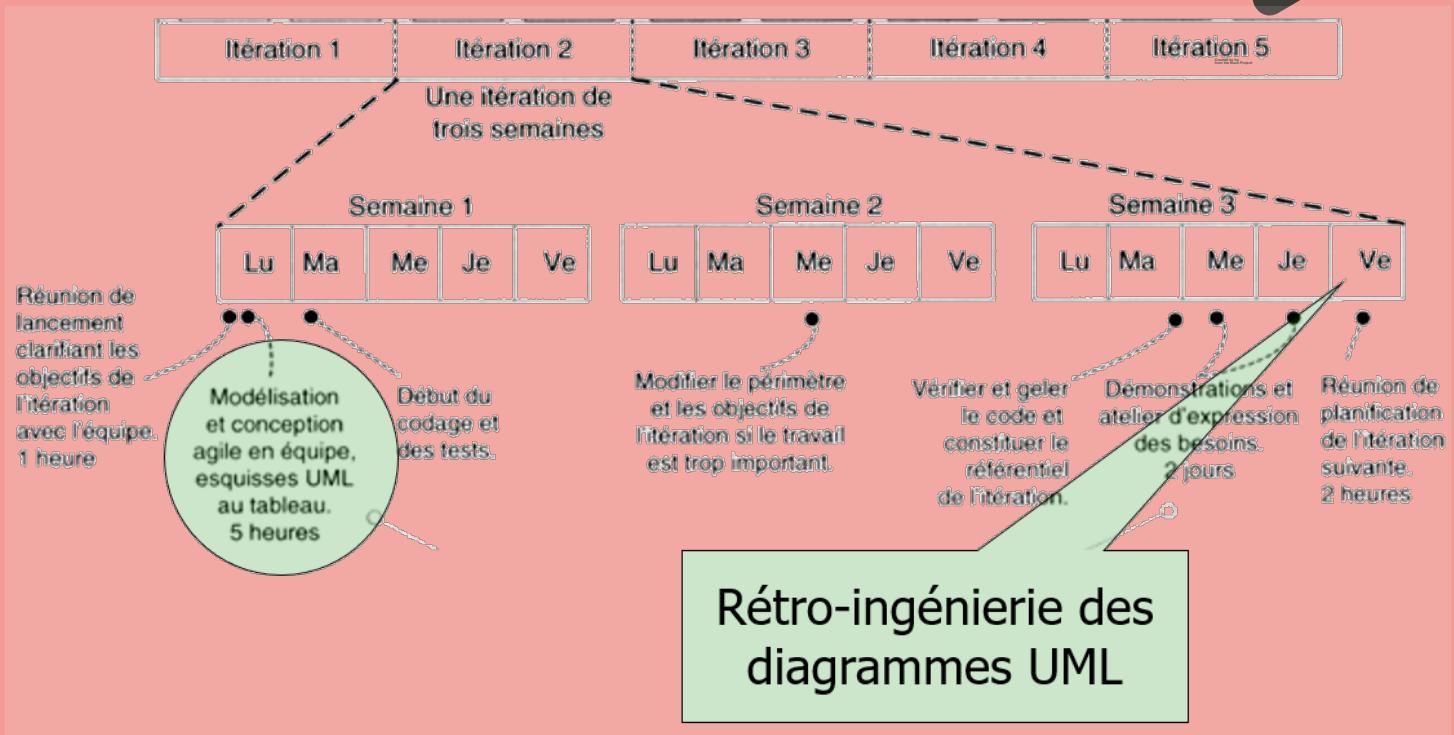
- Papyrus
- Modeliosoft
- PlantUML

UML EN MODE ESQUISSE



- Esquisses murales UML au début d'une itération
- Suivies par 3 semaines de codage et de tests
 - Les modèles évolueront avec le code
- Entamer de nouveau la modélisation de nouvelles esquisses
- Générer les diagrammes à partir du code existant le jours précédent cette modélisation

EXEMPLE D'UNE ITÉRATION



MODULE ANALYSE ET CONCEPTION

ORIENTÉ OBJET

1. Try to fix bug in production
2. Spectre de la conception
3. Analyse et conception orienté objet 21.26m
4. AOO - Approche orienté objet 7.26m
5. Première étude de la conception

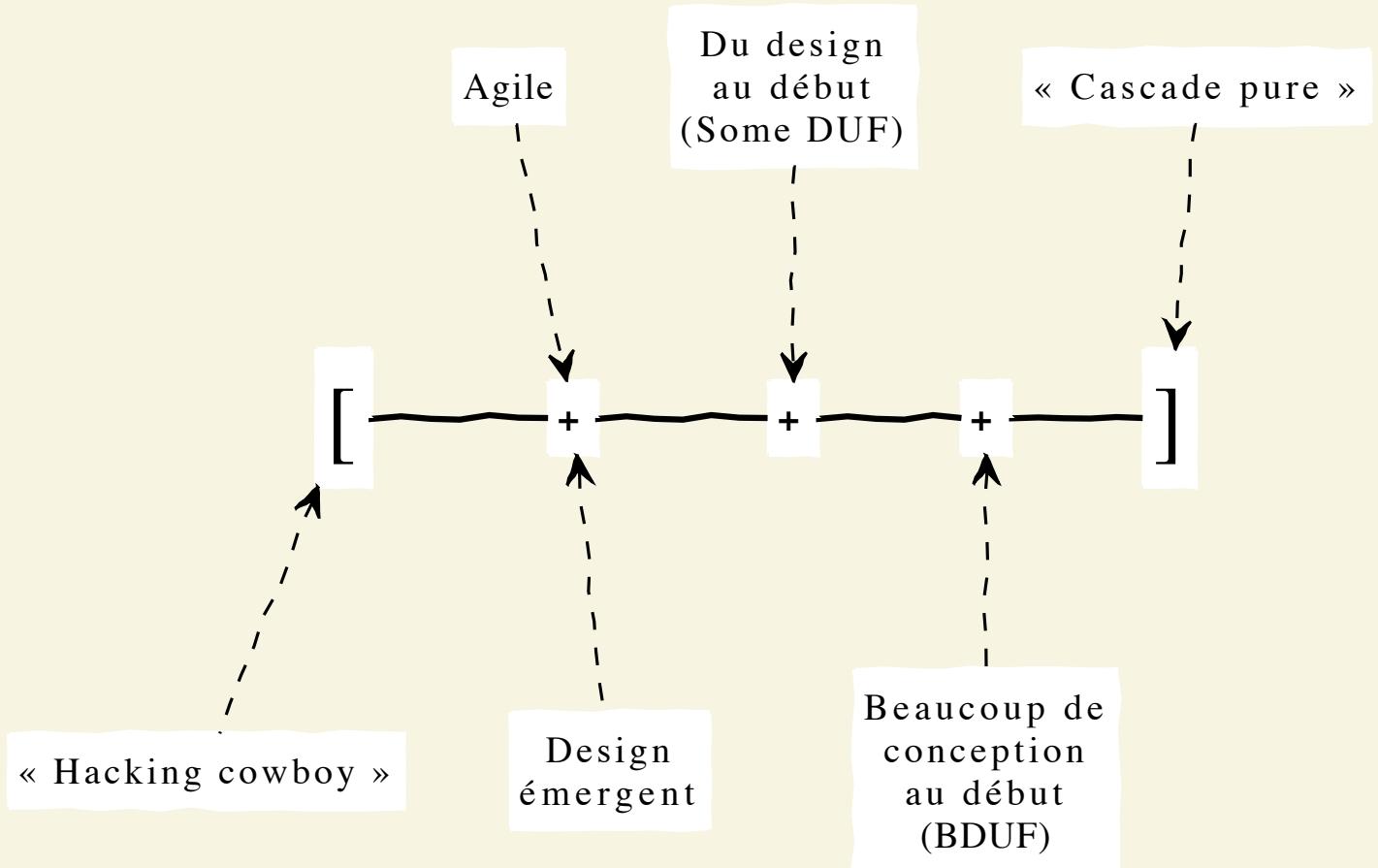
[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexité](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

TRY TO FIX BUG IN PRODUCTION

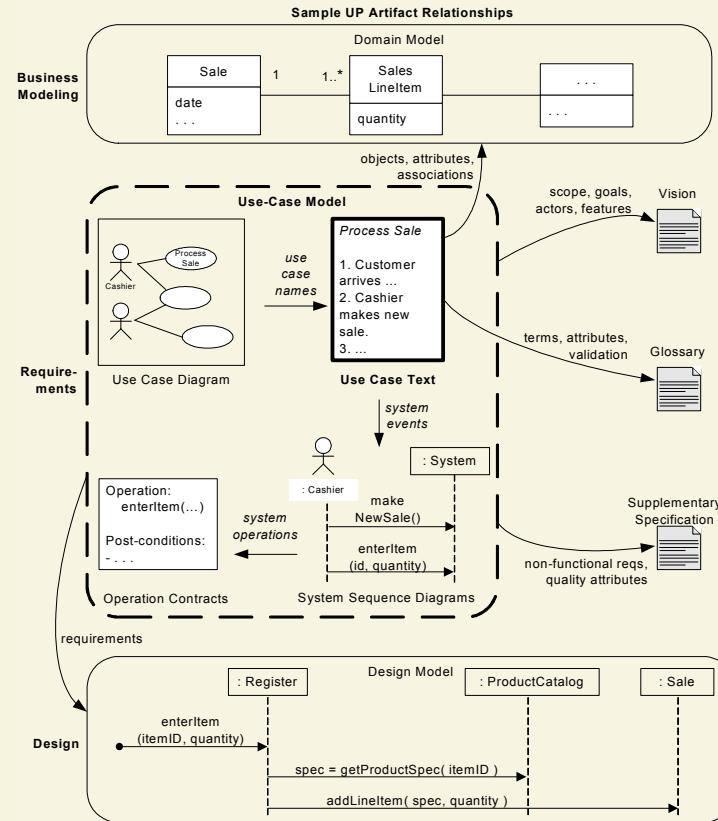
4 . 2

SPECTRE DE LA CONCEPTION

Figure 2.1 des notes de cours.



SURVOL DES MODÈLES



ANALYSE VS CONCEPTION

C'est de la modélisation!

4 . 5

ANALYSE

- Faire un modèle du *problème*
 - Diagramme de classes conceptuelles
 - Modèle du domaine
- Investiger le problème
- Comprendre les besoins
- Ne pas (encore) proposer la solution

CONCEPTION

- Faire un *modèle* d'une *solution*
 - Diagrammes de classes logicielles
 - Diagrammes d'interaction (séquence)
 - etc.
- Ce n'est pas la solution
- Facilite une évaluation des qualités comme la modularité

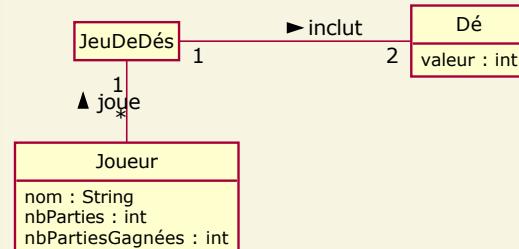
DÉCALAGE DES PRÉSENTATIONS

Comprenez-vous ce que fait ce programme?

```
0001110110001001001001010100001011110011
```

DÉCALAGE DES PRÉSENTATIONS

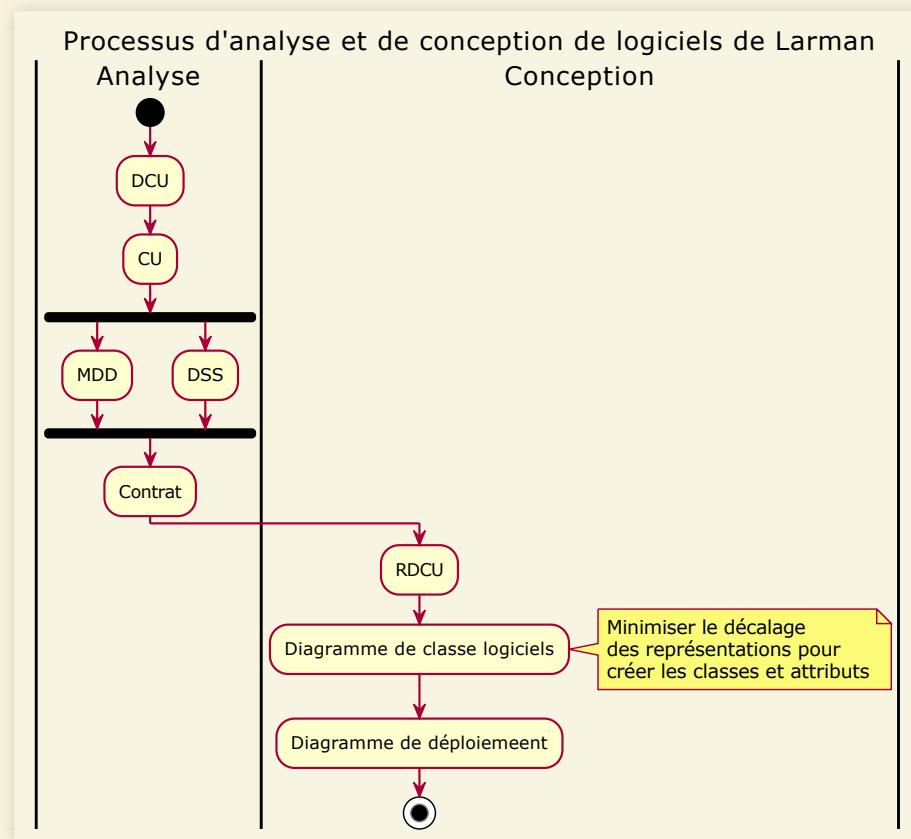
Comprenez-vous ce que fait ce programme?



DÉCALAGE DES PRÉSENTATIONS

- Plus une solution (conception) ressemble à une description du problème, plus elle est facile à comprendre.
- La “distance” entre la représentation d'un problème et la représentation de sa solution s'appelle **le décalage des représentations**.

ANALYSE VERSUS CONCEPTION



LECTURES À FAIRE

Le chapitre 1 des notes de cours.

POURQUOI UNE APPROCHE OO ?

- Objets logiciels sont proche des objets de la vraie vie
 - Structure du programme peut être plus clair
- OO: technologie utilisée en industrie
 - Outils, langages, modèles, etc.
- Objets peuvent être réutilisés dans plusieurs programmes
- Liaison dynamique : modules peuvent être spécifiés à l'exécution

COMMENT DÉVELOPPER EN OO

- Il existe beaucoup de conseils, d'approches
- Approche méthodique (ingénierie)
 - Conception par responsabilité: la modularité
 - GRASP (General Responsibility Assignment Software Pattern)
 - Patterns GoF (Gang of Four)
 - UML (Unified Modeling Language)
 - UP (Unified Process)

UML

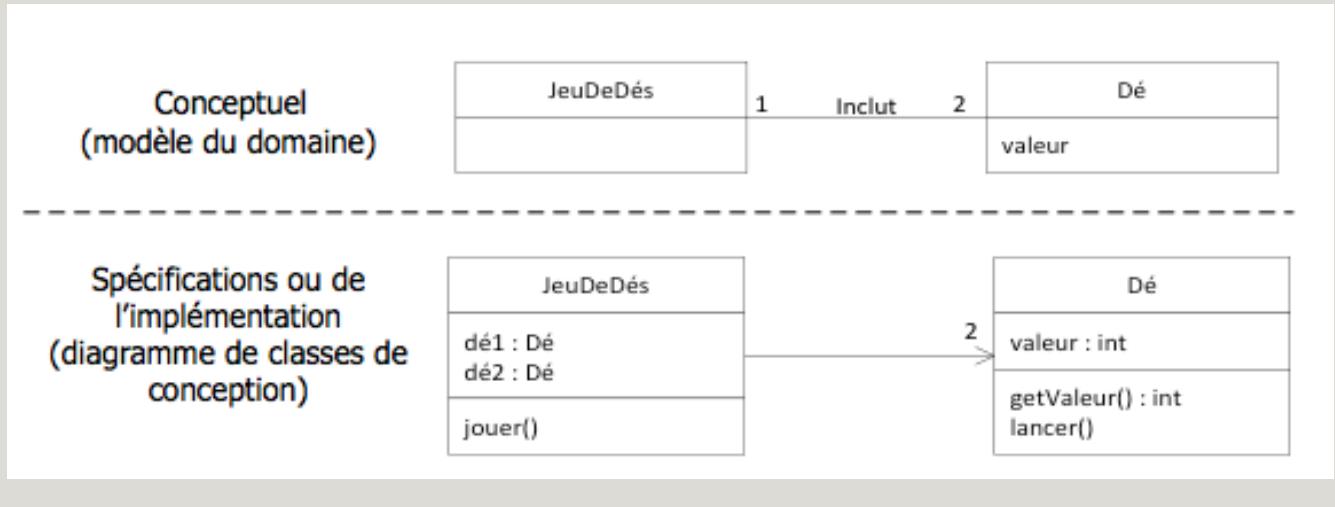


- « Langage visuel dédié à la spécification, la construction et la documentation des artefacts d'un système » – OMG
- Trois façons d'appliquer UML
 - Mode esquisse – diagrammes informels et incomplets, tracés à la main
 - Mode en plan – diagrammes détaillés
 - Comme langage de programmation – spécification complète et exécutable d'un système logiciel en UML

PERSPECTIVES UML



- Conceptuel (monde réel)
- Spécifications (logicielles)
- Implémentation (logicielle)



SPÉCIFICATION DE « CLASSE </>

1. **Classe conceptuelle:** concept ou entité du monde réel.
2. **Classe logicielle:** composant logiciel du point de vue des spécifications ou de l'implémentation, indépendamment du processus ou de la méthode.
3. **Classe d'implémentation:** Classe implémentée dans un langage OO spécifique tel que Java.

VALIDATION DE LA COMPRÉHENSION >

Quels sont les genres des classes dans l'exemple suivant?

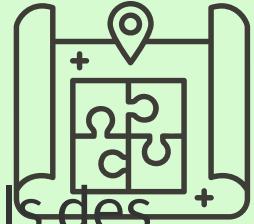
```
public class Avion
{
    private String numéroDeVol;

    public List getHistoriqueDuVol() {...}
}
```



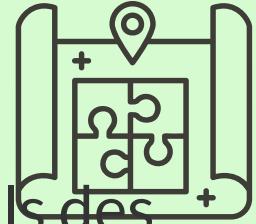
1. Conceptuelle, Spécification, Spécification
2. Implémentation, Conceptuelle, Spécification
3. Implémentation, Spécification, Conceptuelle

CONCEPTION D'OBJET 1/2



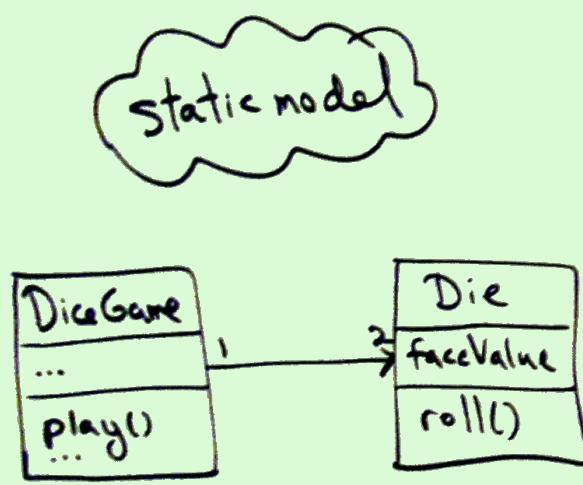
- Comment les développeurs conçoivent-ils des objets?
 - Ils codent. On conçoit en codant (vous avez fait ça au CÉGEP, en INF111, en LOG121, en stage?)
 - Outils de réusinage (refactoring)

CONCEPTION D'OBJET 2/2



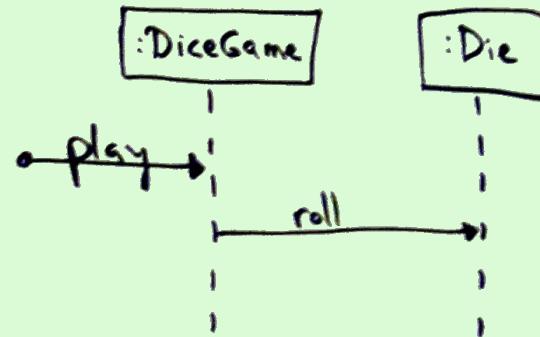
- Comment les développeurs conçoivent-ils des objets?
 - Ils tracent des diagrammes puis ils codent.
 - UML au tableau blanc, puis Java dans Eclipse
- Ils se contentent de tracer des diagrammes.
 - Les outils ne sont pas encore capables de faire le codage à partir des diagrammes... (est-ce vrai?)

QU'EST-CE QUI EST PLUS IMPORTANT?



UML Class Diagram

Dynamic Model



UML Sequence Diagram

CARTES CRC



- Classe, Responsabilité, Collaborateurs
- Utilisées pour faire le design

1. LOG121 (Cay Horstmann, chap: 2.12):

| Mailbox | |
|--------------------------------------|--------------|
| <i>manage passcode</i> | MessageQueue |
| <i>manage greeting</i> | |
| <i>manage new and saved messages</i> | |
| | |

EXAMPLE CRC

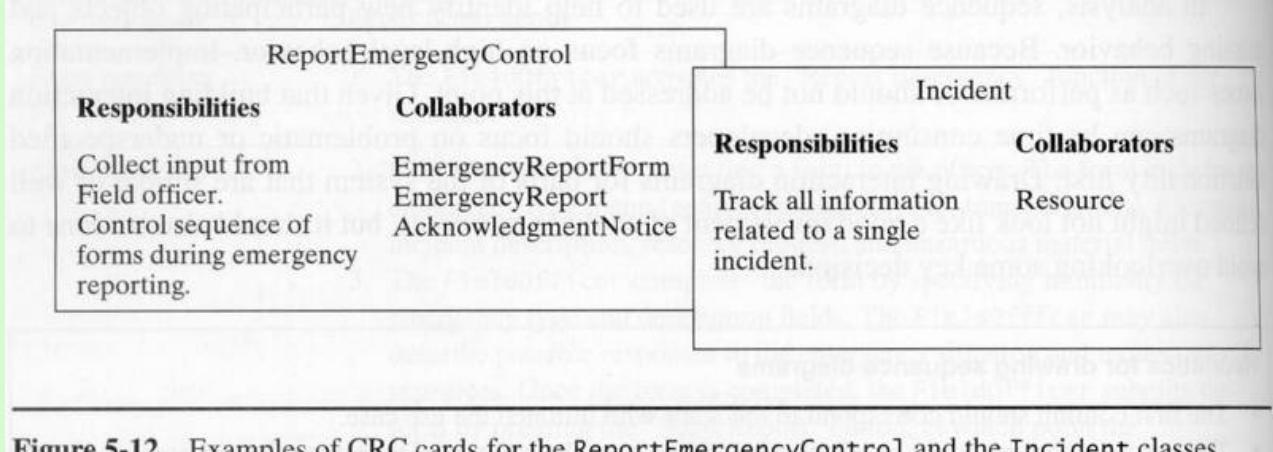


Figure 5-12 Examples of CRC cards for the ReportEmergencyControl and the Incident classes.

ref: Object-Oriented Software Engineering, Bruegge & Dutoit, Prentice-Hall, 2000.

RÉSUMÉ



- Modélisation agile
 - Caméra pour numériser les esquisses
 - Habilité avec UML
- Modélisation dynamique est importante
- Pas beaucoup de temps pour modéliser
 - Une itération de 3 semaines (120 h)
 - 2 à 8 heures de modélisation, pas plus

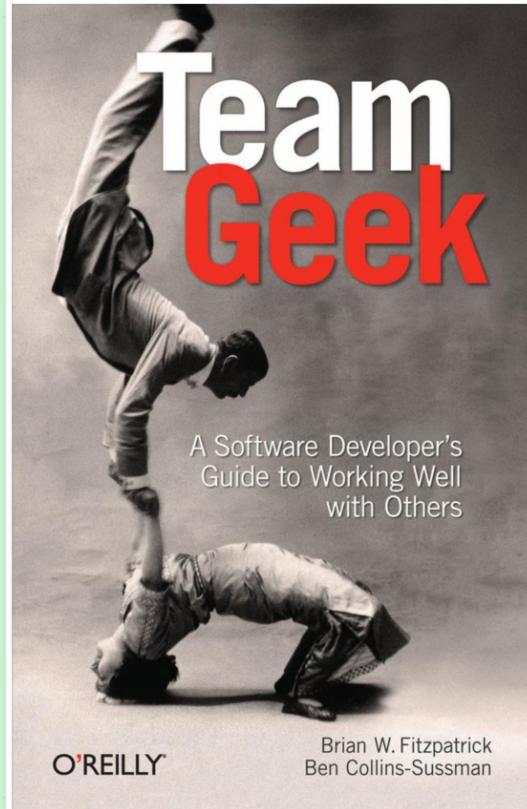
MODULE TRAVAIL D'ÉQUIPE



1. Travail d'équipe (HRC) - 25.53m
2. Confiance - 4.33m
3. Humilité - 10.05m
4. Culture d'équipe - 20.14m
5. Rencontres efficace - 7.17m
6. Équipe en laboratoire

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

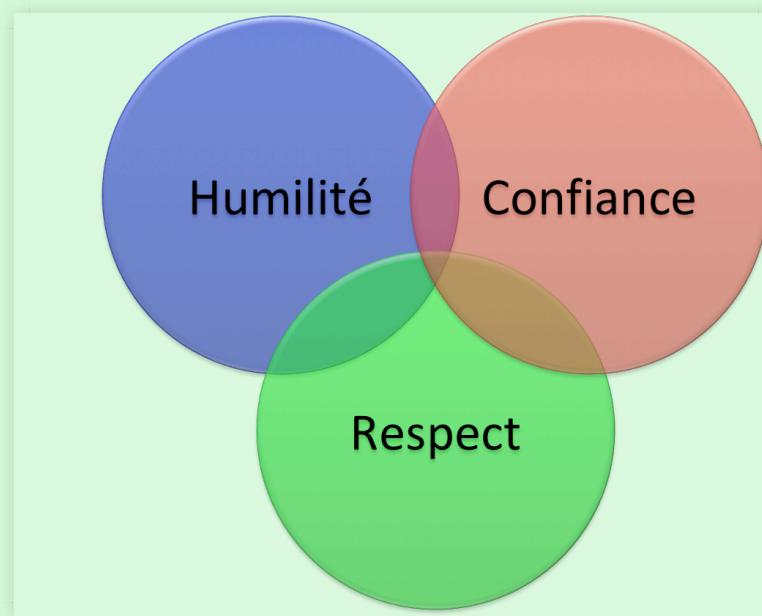
DÉVELOPPEMENT DE LOGICIEL EN ÉQUIPE



5 . 2

DÉVELOPPEMENT DE LOGICIEL EN ÉQUIPE

Pratiquement tout conflit social est dû à un manque d'humilité, de respect ou de confiance.



référence: «Team Geek» Brian W. Fitzpatrick (Google), Ben Collins-Sussman (Subversion, Google)

HUMILITÉ



Trouvez-moi un exemple ou:

- un étudiant ou un professeur fait preuve d'humilité

HUMILITÉ



Trouvez-moi un exemple ou:

- un étudiant ou un professeur fait preuve d'humilité
- un étudiant ou un professeur ne fait pas preuve d'humilité

RESPECT



Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier

RESPECT



Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier
- envers les élèves de la classe

RESPECT



Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier
- envers les élèves de la classe
- d'un étudiant envers un professeur ou un chargé de laboratoire

RESPECT



Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier
- envers les élèves de la classe
- d'un étudiant envers un professeur ou un chargé de laboratoire
- d'un professeur/chargé de laboratoire envers un étudiant

CONFiance



Trouvez-moi un exemple d'une:

- action qui donne confiance à mes coéquipiers

CONFiance



Trouvez-moi un exemple d'une:

- action qui donne confiance à mes coéquipiers
- action qui diminue la confiance envers mes coéquipiers

POURQUOI L'ENSEIGNANT DÉCIDE LES MEMBRES DE L'ÉQUIPE? (1/2)

- Pour apprendre les compétences non-techniques
 - Demandées par les employeurs
 - N'évoluent pas autant que les compétences techniques (p.ex. Flash)
- On ne choisit pas (facilement) son équipe en entreprise
- Travailler avec des collègues différents tout au long de son baccalauréat élargit son réseau de contacts
- Impact positif pour la carrière



Mark Granovetter, «The Strength of Weak Ties », Academic Press, 1977.

POURQUOI L'ENSEIGNANT DÉCIDE LES MEMBRES DE L'ÉQUIPE? (2/2)

- Favorise le « coaching » car il y a de la diversité
- Un coéquipier moins fort dans un aspect du projet peut bénéficier du coaching d'un membre plus fort
- Le « coach » peut bénéficier également de ce qu'on appelle «deep learning»
- L'équipe est finalement plus cohésive et plus performante



ÉVITER LES COMPORTEMENTS DYSFONCTIONNELS



- Chacun fait “sa part” sans se soucier de la santé de l’équipe et de la qualité globale du projet
 - L’équipe > moi (appliquer l’humilité)
 - Rester flexible pour les tâches, s’adapter
 - Proactivité envers problèmes dans l’équipe:
 - Exiger la présence des coéquipiers
 - Demander de l’aide (technologies)
 - S’impliquer!

ÉVITER LES COMPORTEMENTS DYSFONCTIONNELS



- Un ou deux membres prennent tout le contrôle (manque de confiance)
 - Reconnaître les niveaux différents et aider (faire du coaching)
 - Être patient avec les membres ayant moins d'expérience
 - Membres ayant besoin d'aide doivent la chercher tôt (être proactif, s'impliquer)
 - Facteur de bus > 1

CONFiance



Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DEL
Couche | Exercice | Typescript | Complexite | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité |
Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

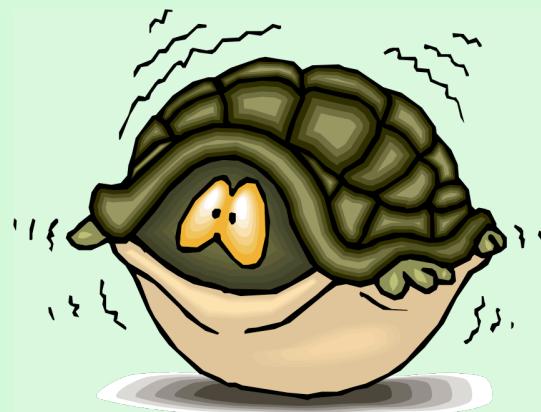
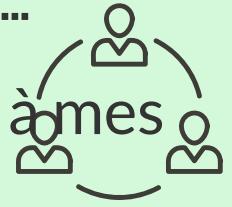
CONFiance

- Je crois que les autres coéquipiers sont compétents et qu'ils feront la bonne chose.
- Je suis à l'aise lorsqu'ils prennent le volant, le cas échéant.



TENDANCES COMPORTEMENTALES CHEZ LES DÉVELOPPEURS...

- Je ne veux pas montrer mon code source à mes coéquipiers...
 - Mon code n'est pas fini...
 - J'ai trop peur d'être jugé(e)...
 - J'ai peur que quelqu'un vole mon idée...
- Il s'agit de l'insécurité. C'est normal!



5 . 13

CACHER SES IDÉES AUGMENTE LE RISQUE...



- de faire des erreurs dans la conception initiale.
- de « réinventer la roue ».
- de terminer le travail plus tard que son compétiteur, qui, lui, a collaboré avec son équipe.

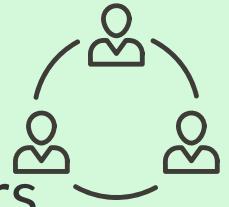


HUMILITÉ



Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DCL
Couche | Exercice | Typescript | Complexite | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité |
Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

HUMILITÉ



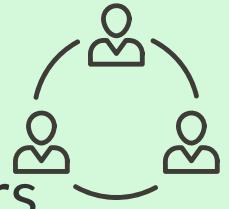
- Je ne suis pas le centre de l'univers.
- Je ne suis ni omniscient ni infaillible

HUMILITÉ



- Je ne suis pas le centre de l'univers.
- Je ne suis ni omniscient ni infaillible
- Je suis ouvert à m'améliorer

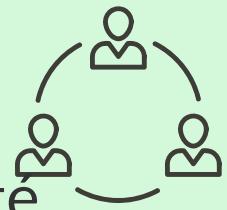
HUMILITÉ



- Je ne suis pas le centre de l'univers.
- Je ne suis ni omniscient ni infaillible
- Je suis ouvert à m'améliorer

Moi < Équipe

HUMILITÉ EN PRATIQUE...



- Si quelqu'un travaille sans assez d'humilité
 - «Voux-tu descendre de ton piédestal?»
- Même si tu es le plus fort en JavaScript dans l'équipe, ne le mets pas en évidence constamment. C'est agaçant.
- Dans certaines cultures, l'humilité est très importante (p. ex. le confucianisme).

HUMILITÉ EN PRATIQUE...



- Apprendre à donner et à accepter les critiques **CONSTRUCTIVES**
- Les donner avec respect, p.ex. « J'ai de la misère à comprendre le flot de contrôle ici dans ton code » plutôt que « Ton code est mal écrit. »
- Les accepter avec humilité, p.ex. « Je comprends ton point de vue. Je vais refactoriser ce code en fin de semaine. »
- Savoir que son estimate de soi n'équivaut pas à sa qualité de code.

HUMILITÉ EN PRATIQUE...



- Apprendre à être patient
- Chacun a sa personnalité et donc sa façon de déboguer, de concevoir, d'écrire du code
- Rester susceptible à l'influence des autres...
- Plus on est ouvert à être influencé, plus on peut influencer
- Plus on est vulnérable, plus on a l'air fort
- Si on veut être entendu, on doit d'abord écouter

CULTURE D'ÉQUIPE



Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DCL
Couche | Exercice | Typescript | Complexite | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité |
Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

CULTURE D'ÉQUIPE



- Qu'est-ce qu'une culture d'équipe?
 - Ensemble de valeurs, d'objectifs, d'expériences qui est unique à chaque équipe.
 - Éléments « codage » : revues de code, développement piloté par les tests, documentation de la conception, etc.
 - Éléments sociaux: sushi à midi, ou un 5 à 7 le vendredi, etc.
- Qu'elle soit bonne ou mauvaise, la culture existera
- Le leader ne décide pas la culture; il s'en occupe.



CULTURE FORTE D'ÉQUIPE



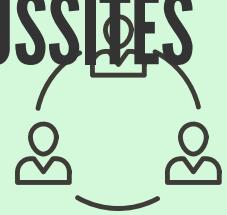
- Culture ouverte au changement qui l'améliore, mais résistant à un changement radical qui lui fait mal.
- Celle qui concentre l'effort sur **la livraison de logiciel génial** est la mieux réussite.
- Efforts pour souder une équipe ne mènent pas toujours à la productivité:
 - faire la fête, surenchère de programmation, faire des rencontres

CULTURE FORTE D'ÉQUIPE



- Une culture forte sera une culture d'auto-sélection
- Culture de code propre et facile à maintenir attirera des développeurs appréciant ces valeurs...
- Culture d'agressivité, d'initiation, de dérapages verbaux, etc. attirera des développeurs appréciant ces valeurs...

ÉLÉMENTS POUR LES CULTURES D'ÉQUIPE RÉUSSITES



- Énoncé de mission d'équipe
- Culture de HRC
- favorise une participation des coéquipiers extrovertis et introvertis
- team.égo > coéquipier[i].égo
- La critique constructive
 - facile à recevoir
 - difficile à donner : ne pas oublier « respect »
- Bonne communication est essentielle
- Moyens de communication diversifiés
- Listes de diffusion
- Documentation de design
- Communication synchrone (rencontres) vs asynchrone (courriel)

COMPORTEMENTS MENACANT UNE BONNE CULTURE

- Ne pas respecter le temps des autres
- Ne pas respecter une décision prise par l'équipe
- Ne pas écouter ou respecter les autres
- Ne pas faire de compromis
- Être perfectionniste
- Être provocateur (troll) / Répondre aux provocateurs (trolls)
- Devenir trop affectif



L'attention et la concentration sont primordiales.

COMMENT AGIR FACE À CES COMPORTEMENTS

- Chercher des faits dans le drame
- Si quelqu'un se plaint, même avec trop d'émotion, lui donner le bénéfice du doute et chercher les causes (malgré le manque de respect, etc.)
- Amener la discussion sur un plan technique si possible.
- Souvent il y a des choses à améliorer dans la situation.
- La gentillesse peut chasser les trolls en fin de compte...



5 . 26

COMMENT AGIR FACE À CES COMPORTEMENTS



- Se concentrer sur l'objectif à long terme
- Un témoin de comportement délétère se demande:
 - Malgré la perte de concentration de l'équipe à court terme, une résolution du drame sera-t-elle bénéfique à l'équipe à long terme?
 - Est-ce que la situation se résoudra d'elle-même?
- Si la réponse est « non » à une de ces questions, mettre fin au comportement immédiatement (sans résolution).



COMMENT AGIR FACE À CES COMPORTEMENTS



- Savoir quand abandonner
- Parfois le comportement d'un coéquipier ne s'améliore pas malgré beaucoup d'efforts. Il faut dans ce cas l'isoler de l'équipe.

Avant de demander un changement d'équipe, il faudra avoir essayé une approche HRC et être en mesure de l'expliquer à l'enseignant.

CULTURE D'ÉQUIPE



- La plupart des gens ne sont pas des imbéciles!
- Cependant, il est naïf de penser aux gens comme « bons » ou « mauvais ».
- La malice qui menace une bonne culture d'équipe est souvent expliquée par l'ignorance, le besoin d'être reconnu, ou un manque d'empathie...
- Il faut toujours être tolérant *envers les gens*, mais ne *pas tolérer les comportements* qui nuisent à une bonne culture d'équipe (selon la norme HRC).

RENCONTRES EFFICACES



Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DEL
Couche | Exercice | Typescript | Complexite | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité |
Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

5 . 30

RENCONTRES EFFICACES



1. N'inviter que ceux qui doivent absolument être présents.
2. Préparer et distribuer l'ordre du jour bien avant que ça commence.
3. Terminer tôt si les objectifs ont été atteints.
4. Respecter l'ordre du jour.
5. Planifier autour des pauses habituelles (p.ex., à midi, à la fin de la journée).

[TeamGeek]

ÉQUIPE EN LABORATOIRE



Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DEL
Couche | Exercice | Typescript | Complexite | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité |
Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

TRAVAIL D'ÉQUIPE AU NIVEAU DU LABORATOIRE



- Quels sont les problèmes HRC rencontré par votre équipe?
- Quels solutions avez vous apportées?

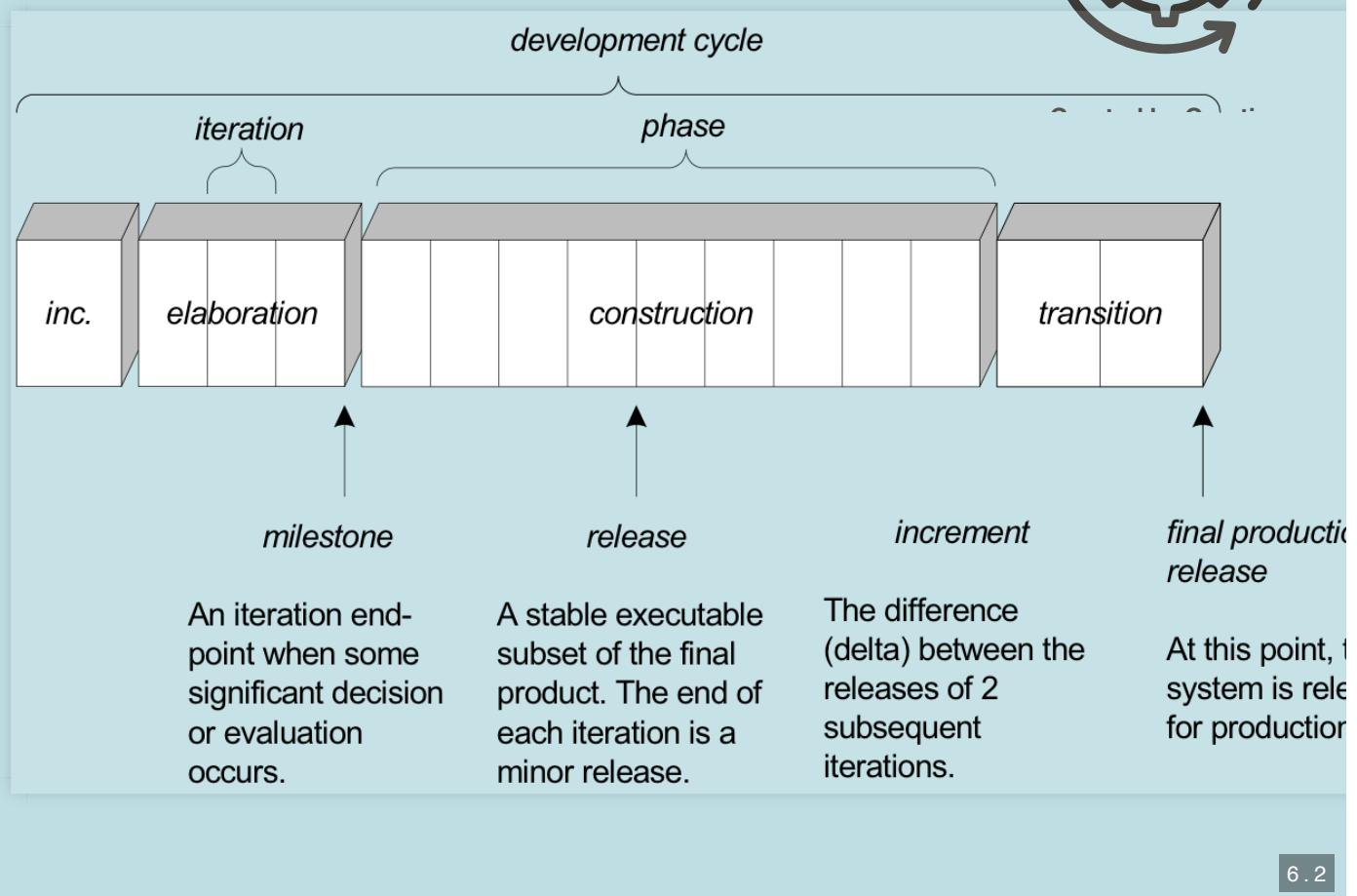
MODULE PROCESSUS UNIFIÉ

1. Phases du processus unifié + inception - 5.21m
2. Durée typique - 15.20m
3. Développement itératif et incrémental - 13.23m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

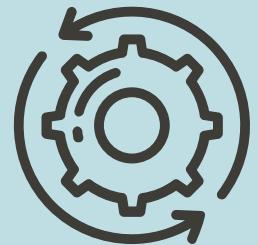
6 . 1

PHASES DU PROCESSUS UNIFIÉ



6.2

PHASE INCEPTION

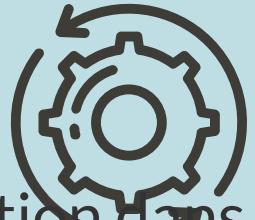


- Étude de faisabilité
 - Vision approximative
 - Estimations globales
 - Cas d'utilisation
 - Continuer ou non (p.ex. chercher le financement)



- Ne pas confondre avec
 - analyse des exigences

PROCESSUS UNIFIÉ?



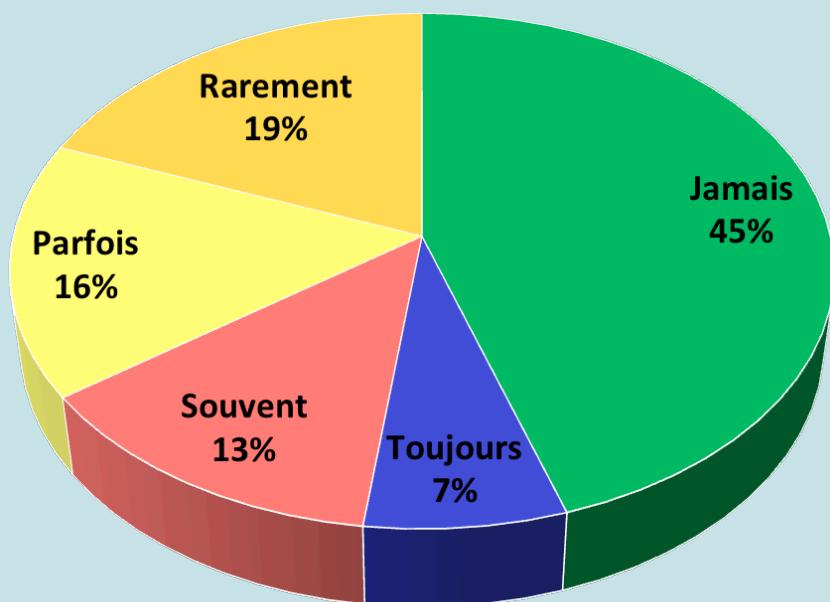
- Quelle est une durée typique d'une itération ~~dans~~ dans le processus unifié?

6.4

APPROCHE ÉVOLUTIVE VS APPROCHE EN CASCADE



Utilisation réelle des fonctionnalités spécifiques dans un processus en cascade



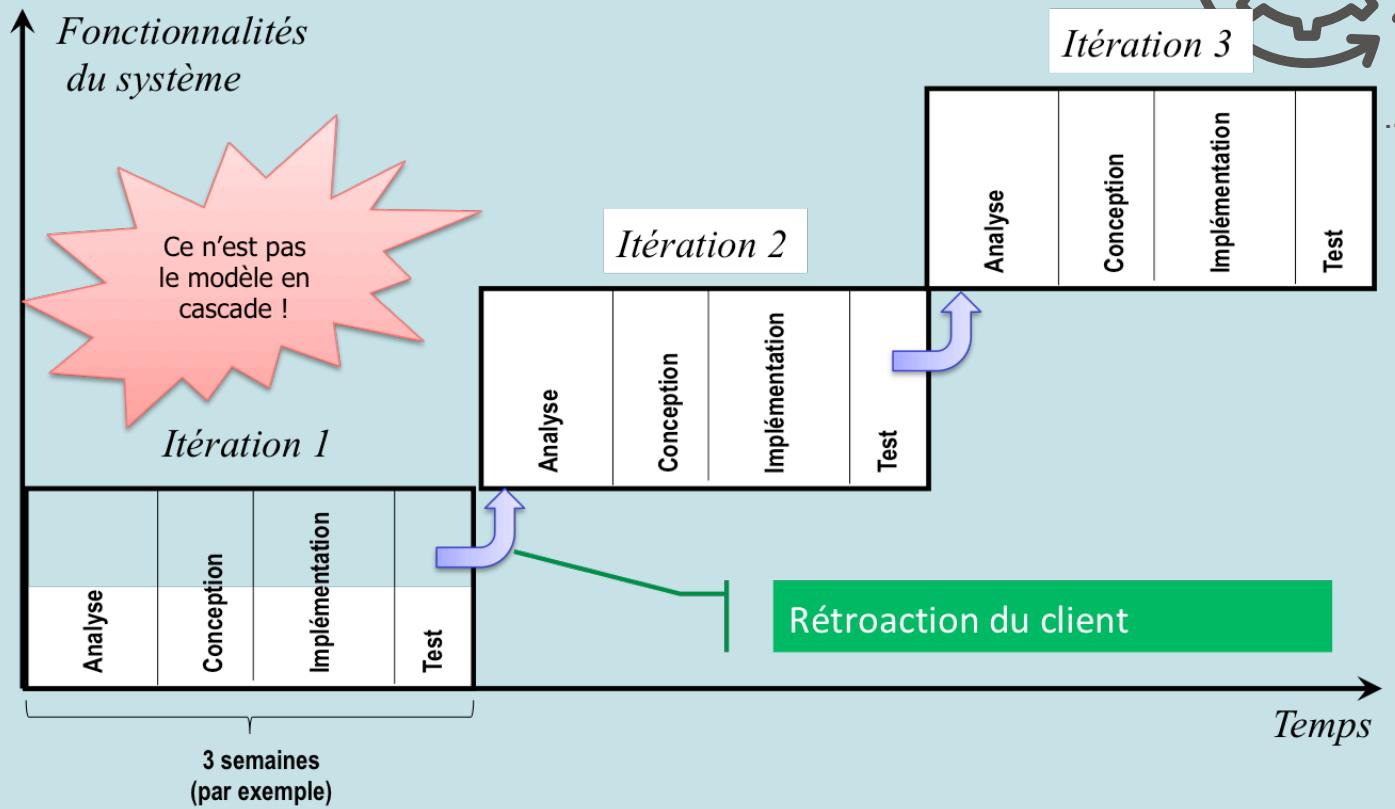
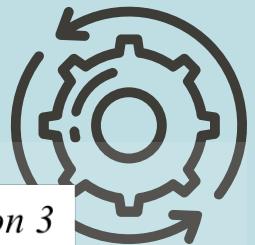
PROCESSUS UNIFIÉ



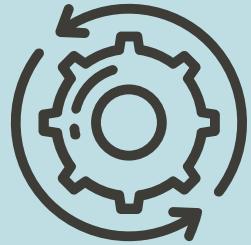
- Souple et ouvert – peut accommoder l'Extreme Programming (XP), Scrum, etc.
- Développement piloté par les tests
 - intégration continue « war room »
 - réunions quotidiennes
 - etc.
- Processus itératif et évolutif

6 . 6

ITÉRATIF ET ÉVOLUTIF



RÉSUMÉ



- Avantages du processus unifié
 - courtes itérations produisant un système partiel
 - rétroaction du client permet d'ajuster la prochaine itération
 - meilleure gestion du risque
- Inconvénients
 - Clients veulent les « promesses » de coûts au début
 - Contrat traditionnel (!) n'est pas adapté pour un processus itératif (orienté modèle en cascade)

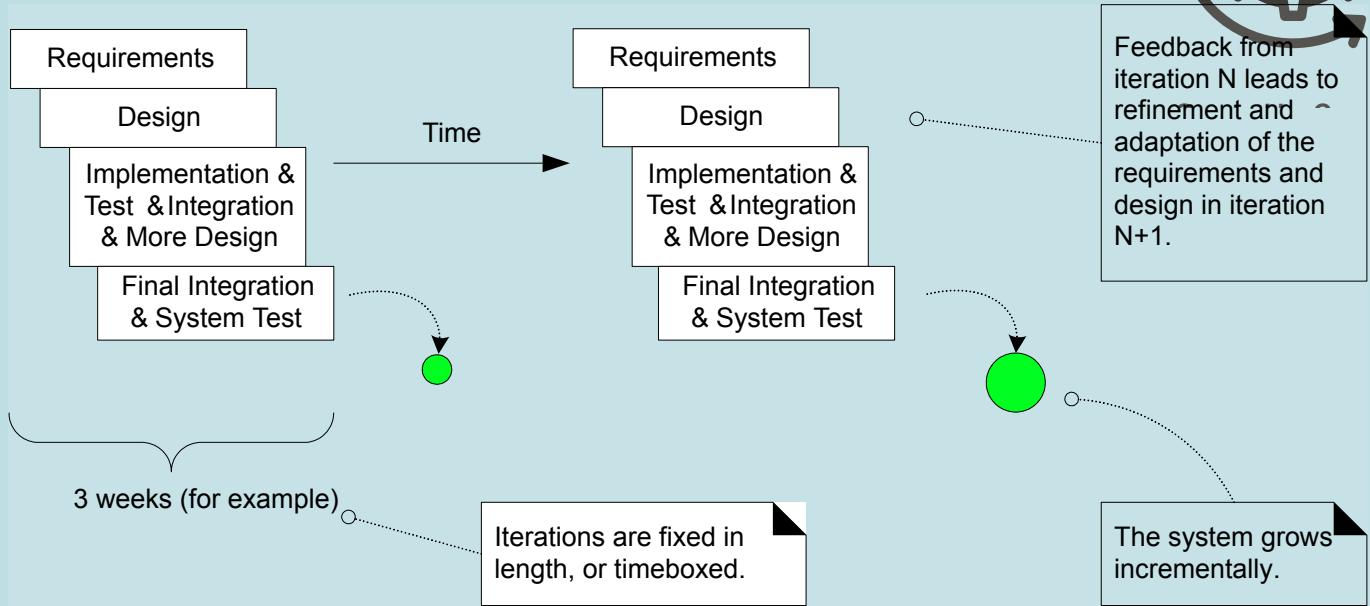
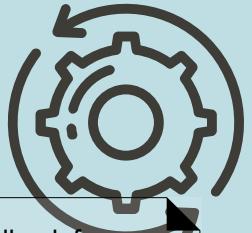
6.8

PROCESSUS UNIFIÉ

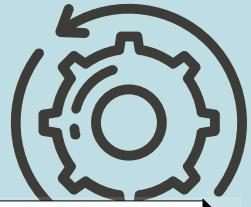


- Souple et ouvert – peut accommoder l'Extreme Programming (XP), Scrum, etc.
- Développement piloté par les tests
- Processus *itératif* et *évolutif*

ITÉRATIF ET ÉVOLUTIF

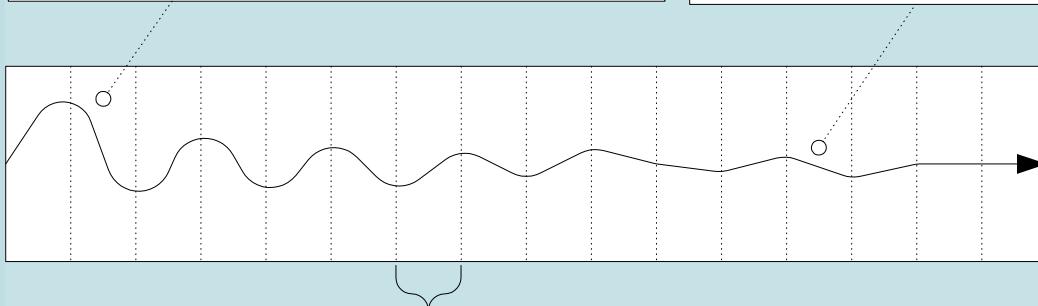


STABILISATION



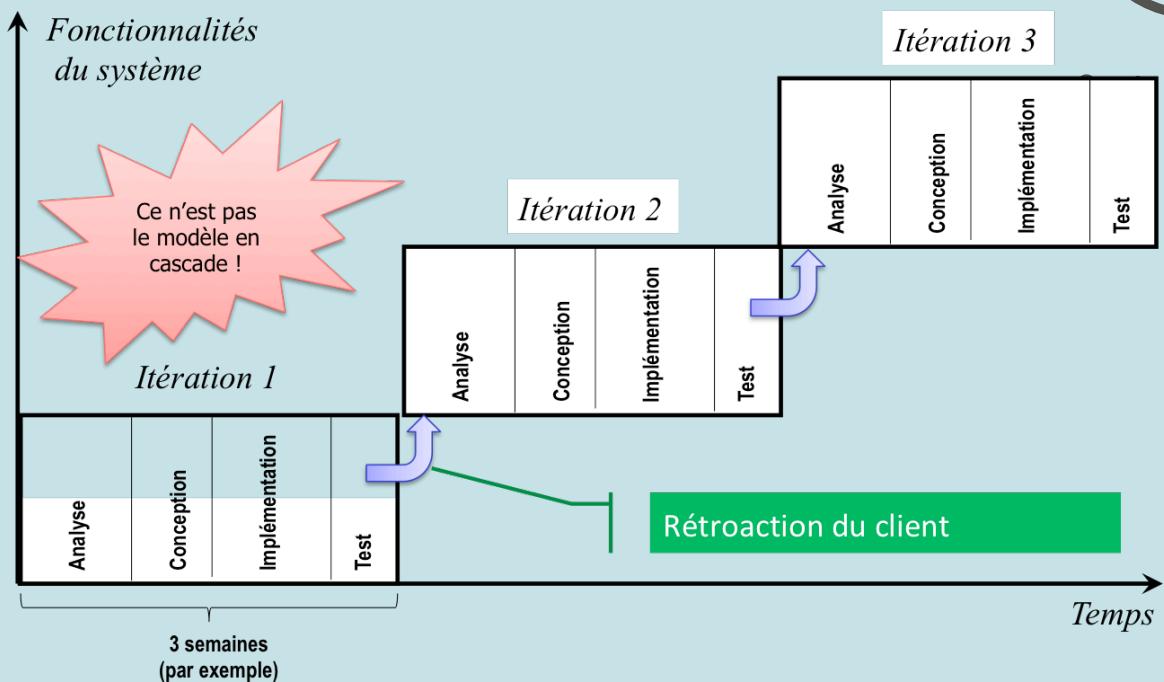
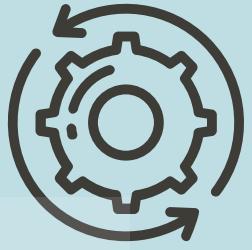
Les premières itérations sont loin de la «bonne voie». Grâce au feed-back et à l'adaptation, le système converge vers les spécifications et la conception les plus appropriées.

Lors des itérations ultérieures, un changement significatif des spécifications est rare mais peut survenir. De telles modifications tardives peuvent procurer à une entreprise un avantage concurrentiel.



une itération : conception,
implémentation, intégration et tests

ITÉRATIF ET ÉVOLUTIF



GESTION DES EXIGENCES (HUMOUR)

Écrire des spécifications correctes et complètes n'est pas facile.



Exact Instructions Challenge - THIS is why my kids hate me. | Josh Darnit



[https://www.youtube.com/embed/cDA3_5982h8?
start=36](https://www.youtube.com/embed/cDA3_5982h8?start=36)

MODULE DIAGRAMME DE CAS D'UTILISATION

1. DCU - Diagramme de cas d'utilisation - 3.24m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

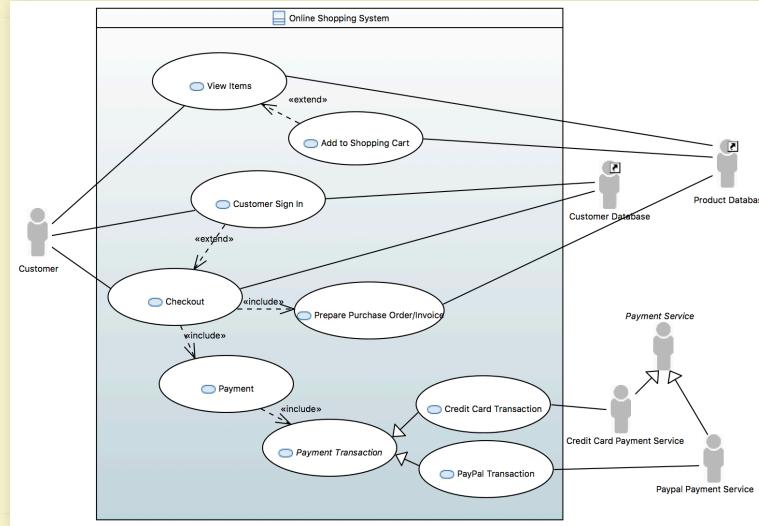
7.1

DCU (LOG410)

DIAGRAMME DE CAS D'UTILISATION



Created by Sven-Peter Ekkebus
from the Noun Project



<https://plantuml.com/use-case-diagram>

CONCEPTS DE BASE DU DIAGRAMME DE CAS D'UTILISATION



Created by Swen-Peter Ekkebus
from the Noun Project

1. Sujet (subject)

- Le sujet est utilisé pour délimiter le système de son environnement

2. Acteur

- Entité externe qui interagit avec le système
- Peut représenter des personnes / utilisateurs et d'autres systèmes externes avec lesquels le système interagit

CONCEPTS DE BASE DU DIAGRAMME DE CAS D'UTILISATION...



Created by Sven-Peter Ekkebus
from the Neun Project

1. Cas d'utilisation

- Fonctionnalité fournie par le système
- Décrit comme une séquence d'étapes / actions

2. Relations

- L'interaction entre les acteurs et un cas d'utilisation est spécifiée à l'aide d'une association.
- Différents types de relations entre les cas d'utilisation: include, extend, generalization/inheritance

TROIS TYPES D'ACTEURS



Created by Sven-Peter Ekkelius
from the Nun Project

1. Acteur principal

- il a un objectif et utilise le système à l'étude (SAE)

2. Acteur de support

- fournit un service, une information, au SAE
- p.ex. système d'autorisation de paiement. Peut être humain, système externe ou équipement.

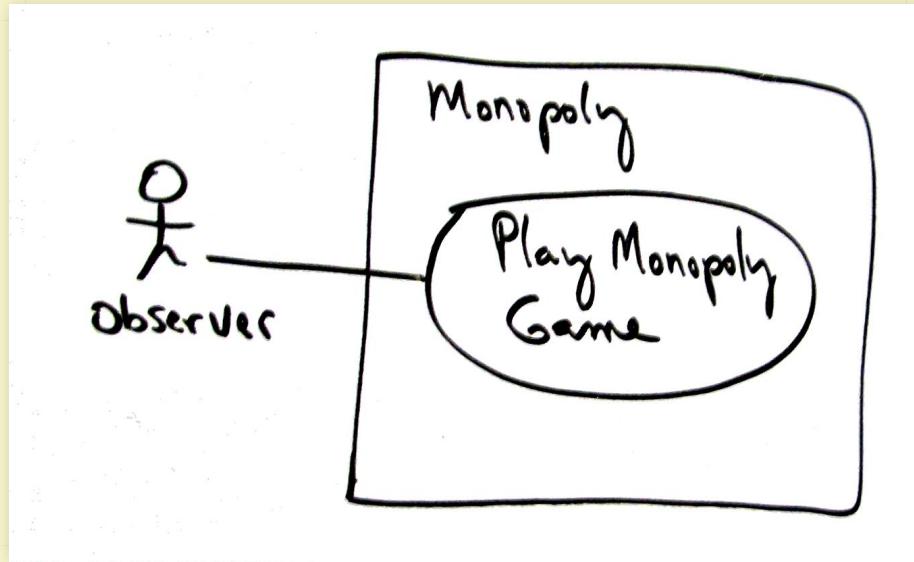
3. Intervenant

- a un intérêt, mais ni acteur principal ou acteur de support (ex: service des taxes du gouvernement, comptable, vérificateur, propriétaire, client, ...)

DIAGRAMME DE CAS D'UTILISATION (AGILE)



Created by Swen-Peter Ekkebus
from the Noun Project

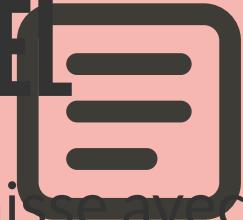


MODULE CAS D'UTILISATION

1. CU - Cas d'utilisation informel - 1.53m
2. CU - Cas d'utilisation - types

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

CAS D'UTILISATION - INFORMÉ



- Created by Adrien Coquet
from the Noun Project
- Traiter une vente.** Un client arrive à la caisse avec les articles qu'il souhaite acheter. Pour enregistrer chaque article, le caissier utilise le système POS, lequel présente le détail des articles et le montant total des achats. Le client fournit les informations nécessaires pour le règlement. Le système valide et enregistre ces informations, puis met à jour les quantités en stock et imprime le ticket de caisse destiné au client. La vente est terminée et le client peut quitter le magasin.

8.2

DE LA DESCRIPTION DU CAS D'UTILISATION AU DIAGRAMME DE CAS D'UTILISATION

4 Basic Flow of Events

1. Customer selects the "Checkout" option
2. If the Customer is not already signed in, then Cmind OSS ask the Customer to sign in – described in **Subflow 6.1**
3. Cmind OSS retrieves Customer information from Customer Database
4. Cmind OSS displays existing shipping address and asks Customer to confirm
5. Customer confirms shipping address or enter a different shipping address
6. Cmind OSS asks Customer to select shipping option
7. Customer chooses a shipping option
8. Cmind OSS prepares purchase order/invoice – described in **Subflow 6.2**
9. Cmind OSS displays the final purchase order/invoice
10. Cmind OSS proceeds with payment – described in **Subflow 6.3**
11. Cmind OSS displays final transaction receipt and send a copy to Customer email address
12. Cmind OSS returns to the Welcome page

6 Subflows

6.1 Customer Sign In – described by the Customer Sign In use case

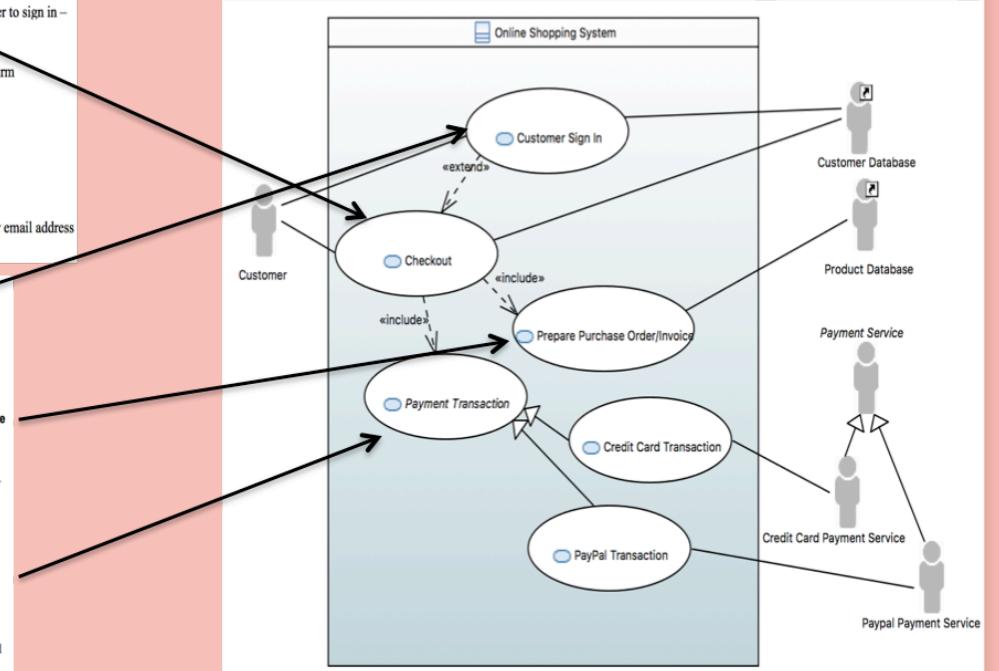
1. Cmind OSS asked Customer to enter username and password
 2. Customer enters username and password
 3. Cmind OSS validates Customer login information with Customer Database
 4. Cmind OSS asks Customer Database to validate Customer Sign In information
- Alternative flow: If Customer login information is invalid, then the use case is terminated*

6.2 Prepare Purchase Order/Invoice – described by the Prepare Purchase Order/Invoice use case

1. Cmind OSS retrieves content of the shopping cart
2. Cmind OSS retrieves product information from Product Database
3. Cmind OSS calculates purchase sub-total (excluding taxes and shipping) based on content of shopping cart and product info
4. Cmind OSS adds shipping cost to sub-total
5. Cmind OSS calculates taxes
6. Cmind OSS calculates purchase total

6.3 Payment – described by the Payment use case

1. Cmind OSS asks Customer to select payment method – described by the Payment use case
2. Cmind OSS asks Customer to select payment method
3. Customer chooses the payment method
4. Cmind OSS displays the existing Customer payment information for the selected method and asks Customer to confirm information
5. Customer confirms payment information
6. Customer makes final purchase confirmation
7. Cmind OSS processes the payment transaction – described by Payment Transaction use case



CAS D'UTILISATION



Created by Adrien Coquet
from the Noun Project

- Que fait le système et non comment
- Trois types:
 - Abrégé
 - Un paragraphe décrivant le scénario principal
 - Informel (casual)
 - Plusieurs paragraphes décrivant plusieurs scénarios
 - Détaillé
 - Toutes les étapes et les variantes sont indiquées en détail, de même que les préconditions et les garanties en cas de succès

DÉFINITIONS



- Acteur
 - Entité qui a un comportement, comme une personne (identifiée par un rôle)
- Scénario
 - Suite spécifique d'actions et d'interactions entre un ou plusieurs acteurs et le système. C'est une histoire particulière de la façon dont on utilise un système
- Cas d'utilisation
 - Collection de scénarios de réussite ou d'échec.

ACTEUR



Created by Adrien Coquet
from the Noun Project

- Quelque chose ayant un comportement
 - un usager, identifié par un rôle
 - un système informatique
 - une organisation
 - par exemple : un caissier



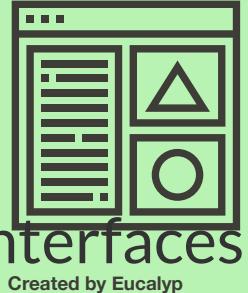
MODULE D'INTERFACE USAGÉ

1. IU - Interface Usager vs DSS 6.22m

Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DCL | Couche | Exercice | Typescript | Complexite | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité | Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

9 . 1

INTERFACE USAGER



- Réaliser les wireframes des différentes interfaces usagées nécessaires pour la réalisation du cas d'utilisation **Noter une réservation**
- Utiliser plant uml
 - <https://plantuml.com/salt>

9 . 2

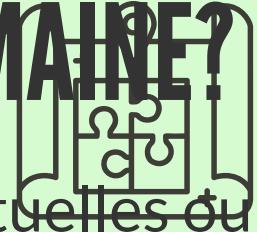
MODULE MDD

1. Modèle du domaine - 12.25m
2. Catégories - 30.27m
3. Classes, Attributs, Associations - 58.47m
4. Composition - 22.47m
5. Révision - 16.05m, 24.12m
6. Affinement du MDD - 25.42m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexité](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

10 . 1

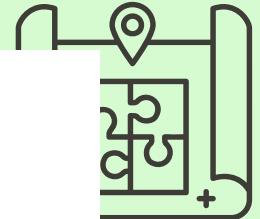
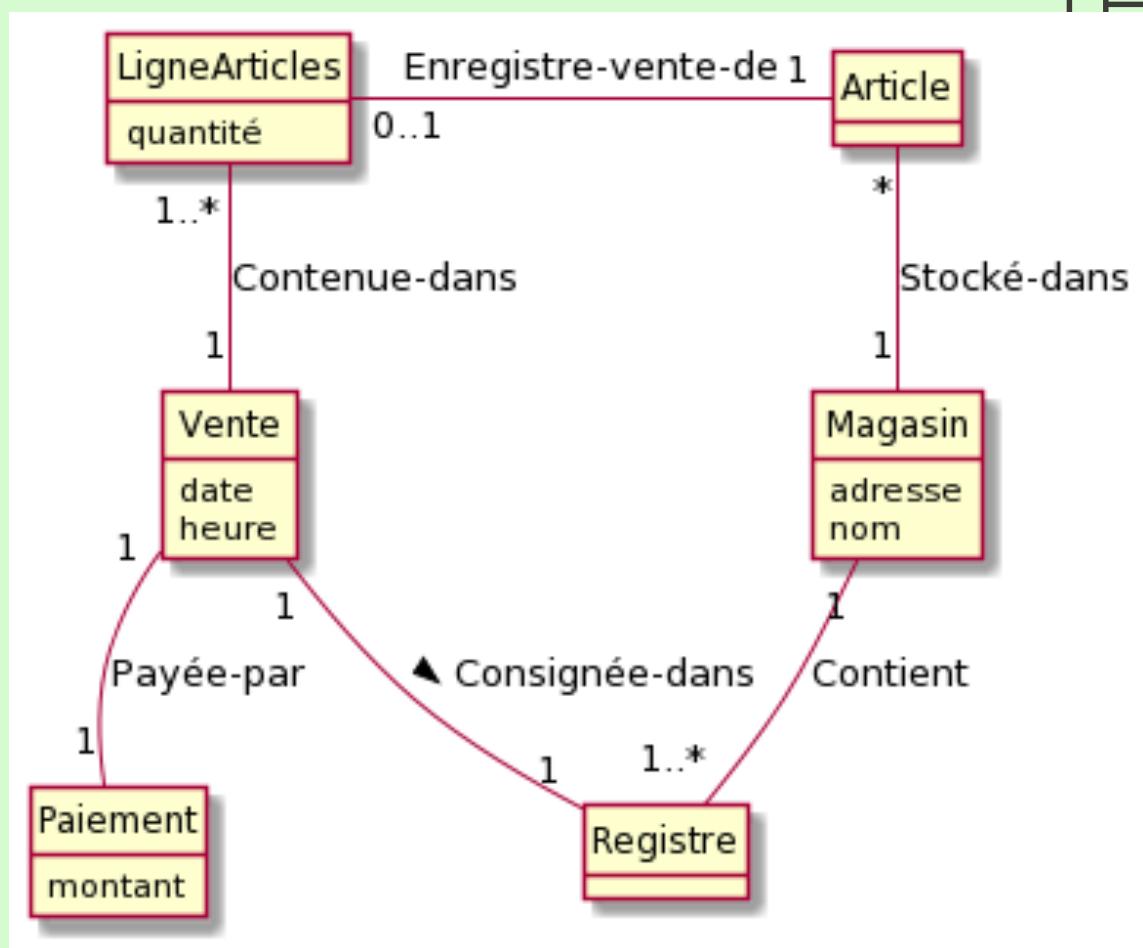
QU'EST-CE QU'UN MODÈLE DU DOMAINE?



Représentation visuelle des classes conceptuelles ou
des objets du monde réel dans un domaine donné
[MO95, Fowler96]

- modèle conceptuel
- modèle objet du domaine
- modèle objet d'analyse

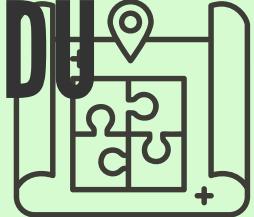
EXEMPLE AVEC L'OUTIL PLANTUML



Plant Uml class diagram

10 . 3

COMMENT CRÉER UN MODÈLE DU DOMAINE?



- Dans les limites des besoins de l'itération en cours:
 1. Identifier les classes conceptuelles.
 2. Les représenter sous forme de classes dans un diagramme UML.
 3. Ajouter des attributs et des associations.

RÉALISER VOTRE PREMIER MDD



En équipe de 2

1. Trouver les classes du domaine
2. Ajouter des associations entre les classes lorsque vous désirez persister cette relation
3. Ajouter un verbe pour décrire cette association
4. Ajouter une multiplicité à l'association

10 . 5

RÉALISER VOTRE PREMIER MDD



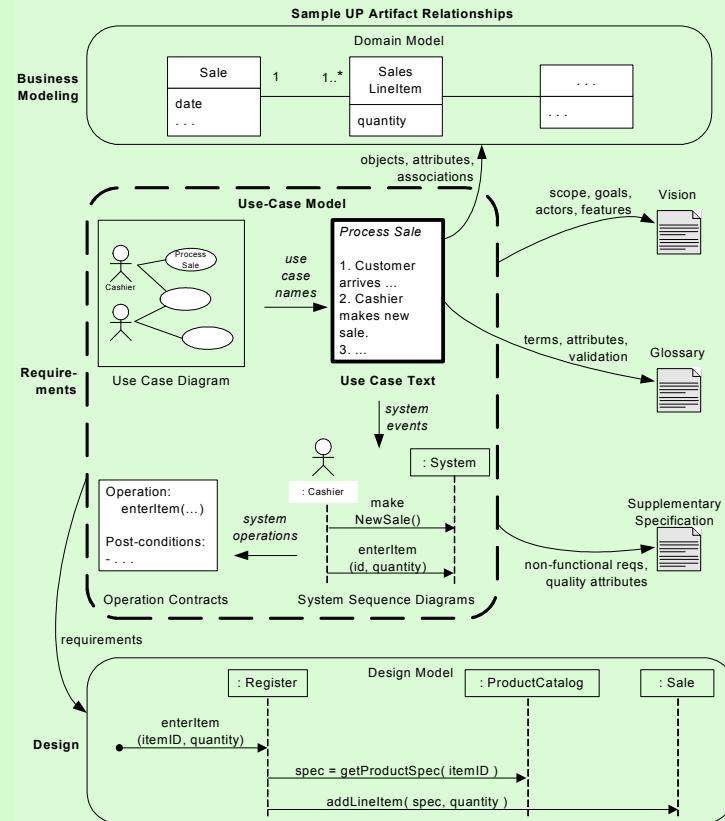
Cas d'utilisation «Noter une réservation»

Scénario principal

1. Un client appelle à l'hôtel pour placer une réservation.
2. Le commis démarre une nouvelle réservation.
3. Le commis entre:
 - La date d'arrivée;
 - La date de départ;
 - Le nom de la catégorie de chambre;
4. Le système inscrit les informations à la réservation.

Interface usager | Réaliser votre premier DSS | Itération suivante

SURVOL VISUEL



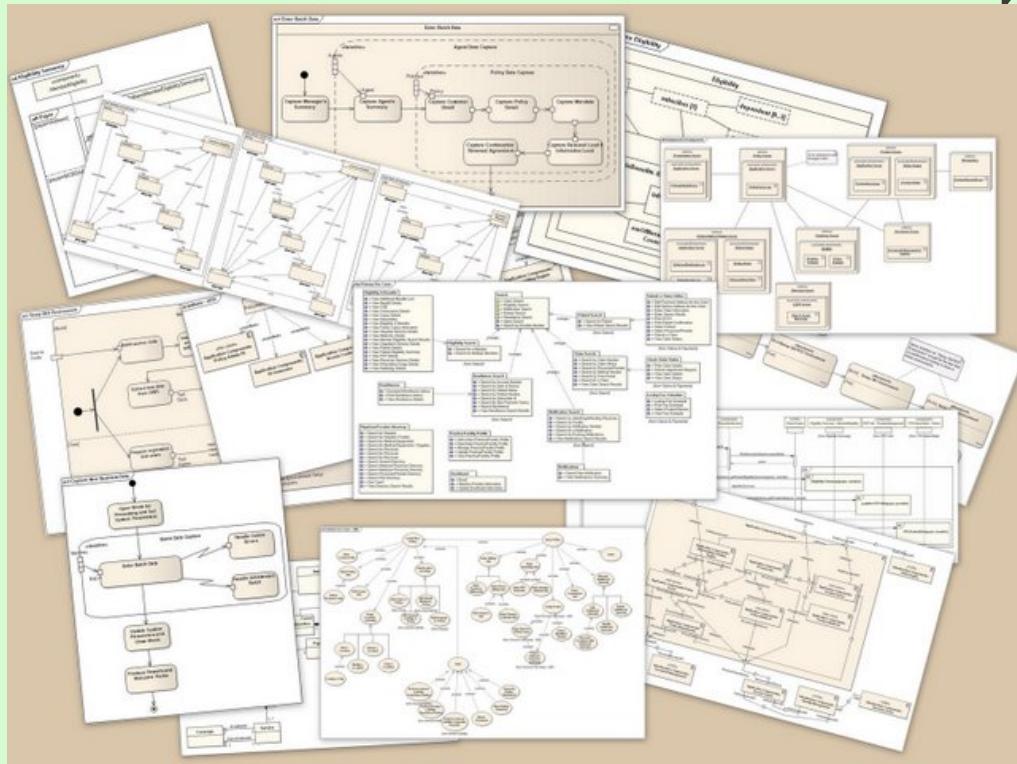
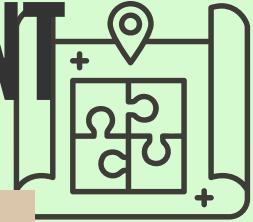
MDD



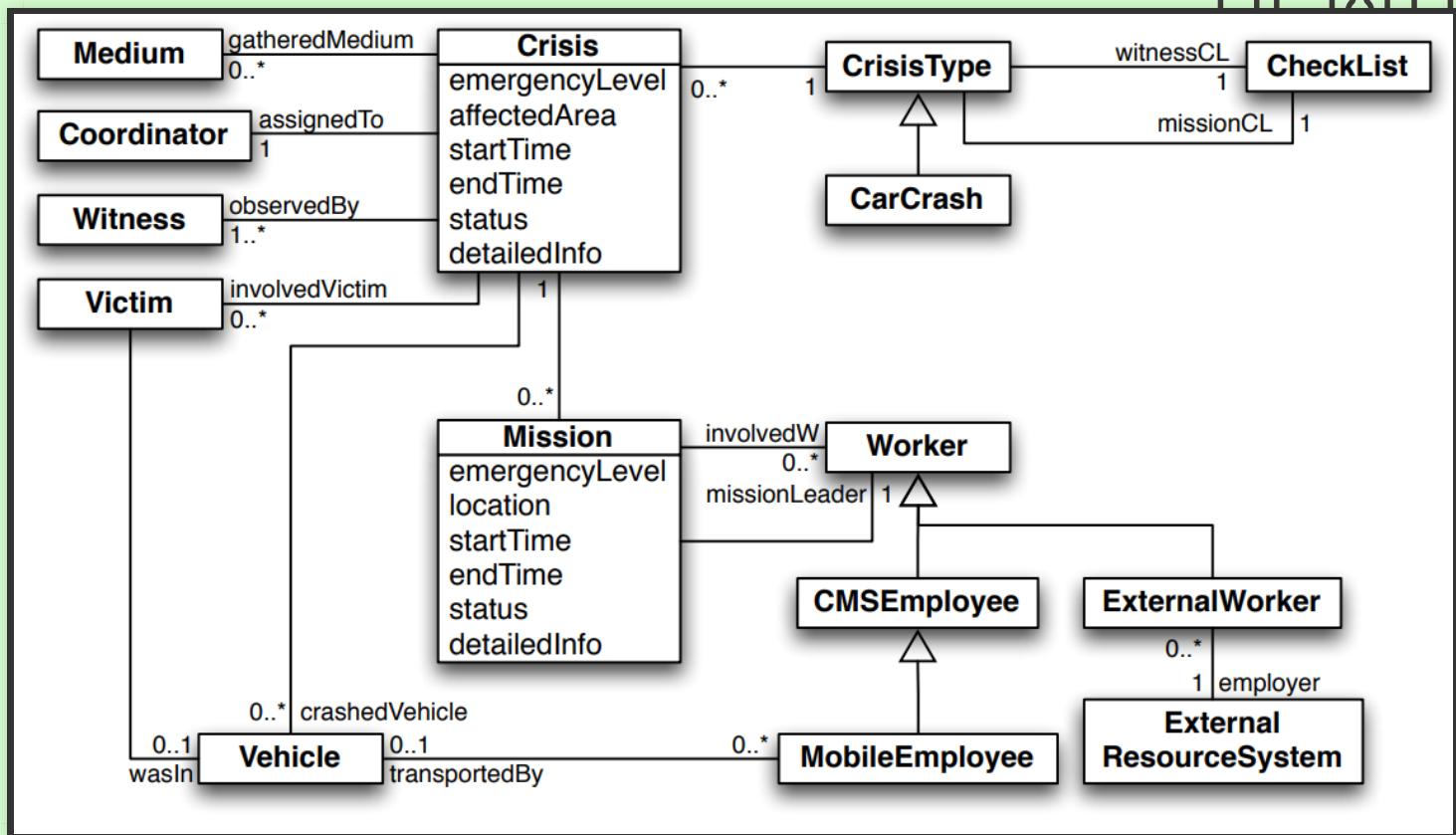
IDENTIFIER CLASSES CONCEPTUELLES

1. Réutiliser modèle existant
2. Catégories de classes
3. Groupes nominaux

MÉTHODE 1: MODÈLE EXISTANT



MÉTHODE 1: MODÈLE EXISTANT



MÉTHODE 2: CHERCHER PAR CATÉGORIE



Section 9.5 du livre.

| | | |
|--|------------------------|--------------------------------------|
| transaction métier | descriptions d'entités | événements notables |
| lignes d'une transaction | catalogues | autres systèmes externes |
| produit ou service lié à une transaction | conteneurs | contenu d'un conteneur |
| où la transaction est-elle enregistrée | Role de personne | objets physiques |
| lieu de la transaction ou service | | documents financiers, contrats, etc. |
| | | instruments financiers |
| | | plannings, manuels, etc. |

CATÉGORIE DE CLASSE

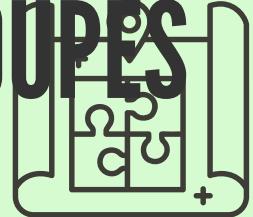


Cas d'utilisation « S'inscrire à un groupe-cours »

1. L'Étudiant commence une inscription.
2. L'Étudiant entre le sigle du cours.
3. Le Système affiche la liste des groupes-cours ainsi que l'horaire de chaque groupe-cours.
4. L'étudiant sélectionne le groupe-cours auquel il veut s'inscrire.
5. ...

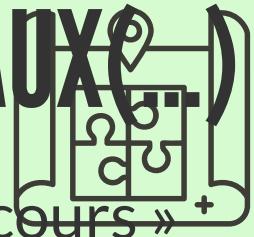
Catégories dans les Notes de cours

MÉTHODE 3: IDENTIFIER LES GROUPES NOMINAUX



- Analyse linguistique du texte des spécifications (Cas d'utilisation, Récit utilisateur, etc.)
- Noms → Classes conceptuelles ou attributs possibles
- Approche *simple*, mais *imprécise*

IDENTIFIER LES GROUPES NOMINAUX



Cas d'utilisation « S'inscrire à un groupe-cours »

1. L'Étudiant commence une inscription.
2. L'Étudiant entre le sigle du cours.
3. Le Système affiche la liste des groupes-cours ainsi que l'horaire de chaque groupe-cours.
4. L'étudiant sélectionne le groupe-cours auquel il veut s'inscrire.
5. ...

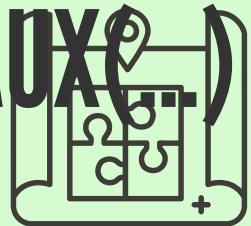
IDENTIFIER LES GROUPES NOMINAUX



Cas d'utilisation «S'inscrire à un groupe-cours»

1. L'Étudiant commence une inscription.
2. L'Étudiant entre le sigle du cours.
3. Le Système affiche la liste des groupes-cours ainsi que l'horaire de chaque groupe-cours.
4. L'étudiant sélectionne le groupe-cours auquel il veut s'inscrire.
5. ...

IDENTIFIER LES GROUPES NOMINAUX



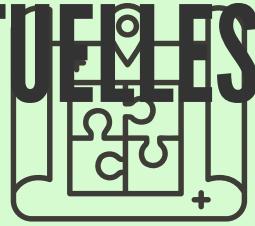
Résultat: *Classes candidates*

- Beaucoup de directives (Chapitre 9)
- Classe vs Attribut?
- Attribut vs Association?

C'est de l'analyse (pas la conception)

EXERCICE: MDD CLASSES CONCEPTUELLES

Google Classrooms Exercices



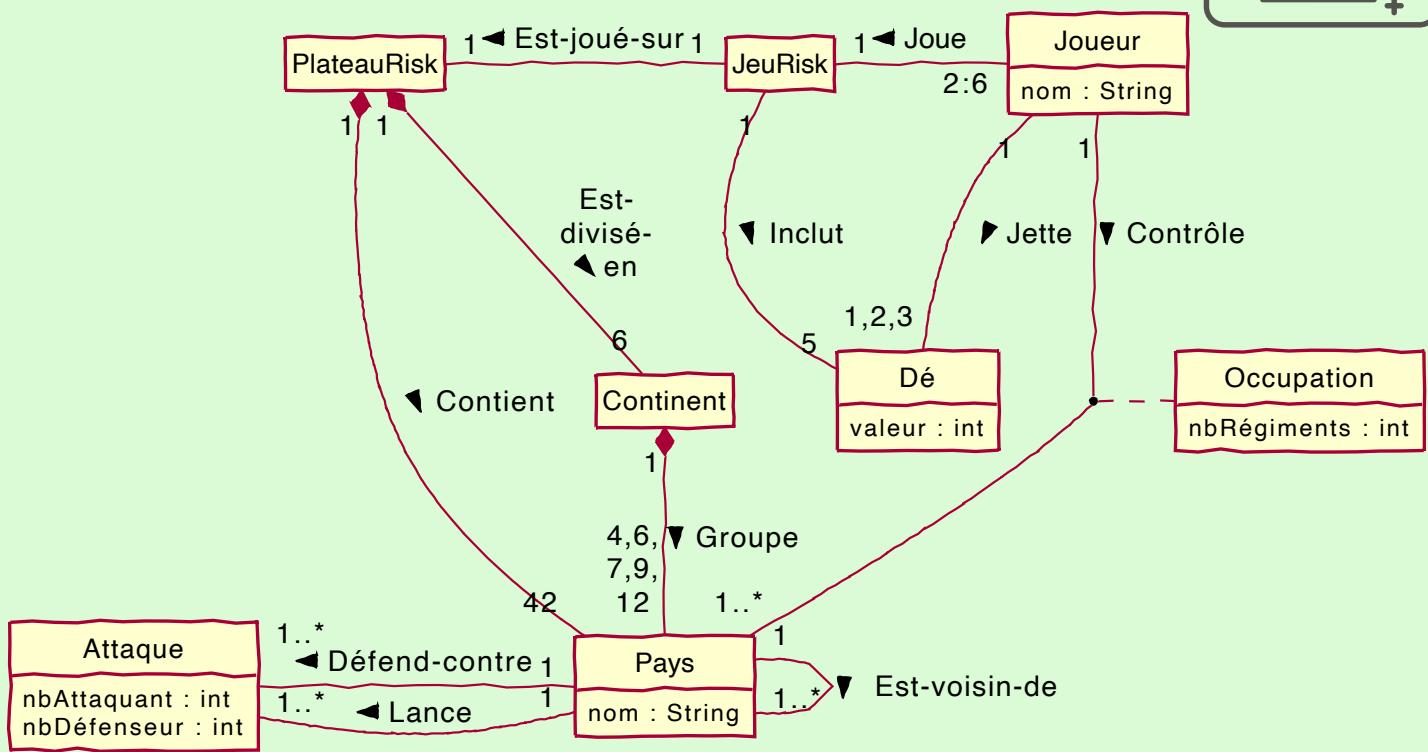
10 . 17

MDD



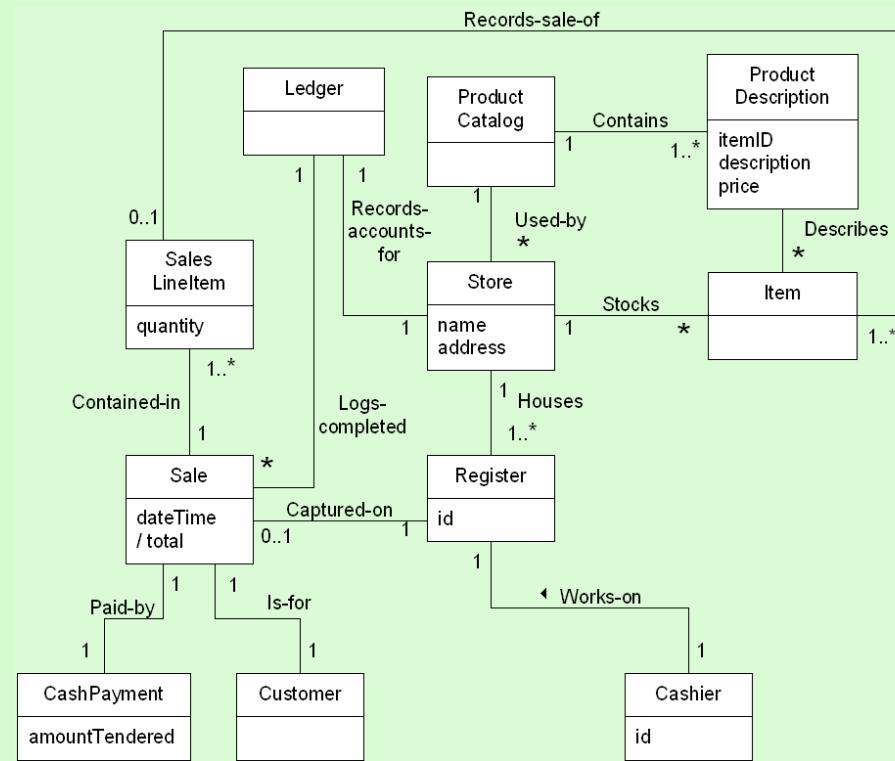
- Classes conceptuelles
- Attributs
- Associations

*MDD - JEU RISK



MDD - ATTRIBUTS NEXTGEN (LARMAN)

Dans LOG210 il **faut** spécifier les types des attributs



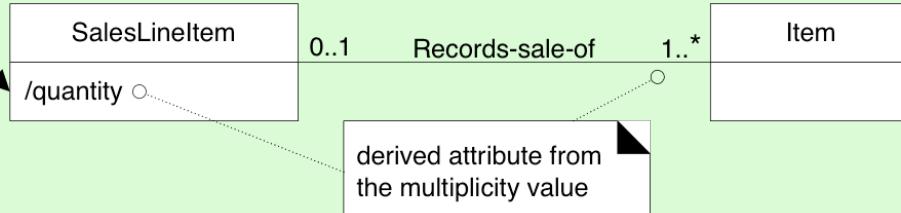
ATTRIBUTS DÉRIVÉS



Each line item records a separate item sale.
For example, 1 tofu package.



Each line item can record a group of the same kind of items.
For example, 6 tofu packages.

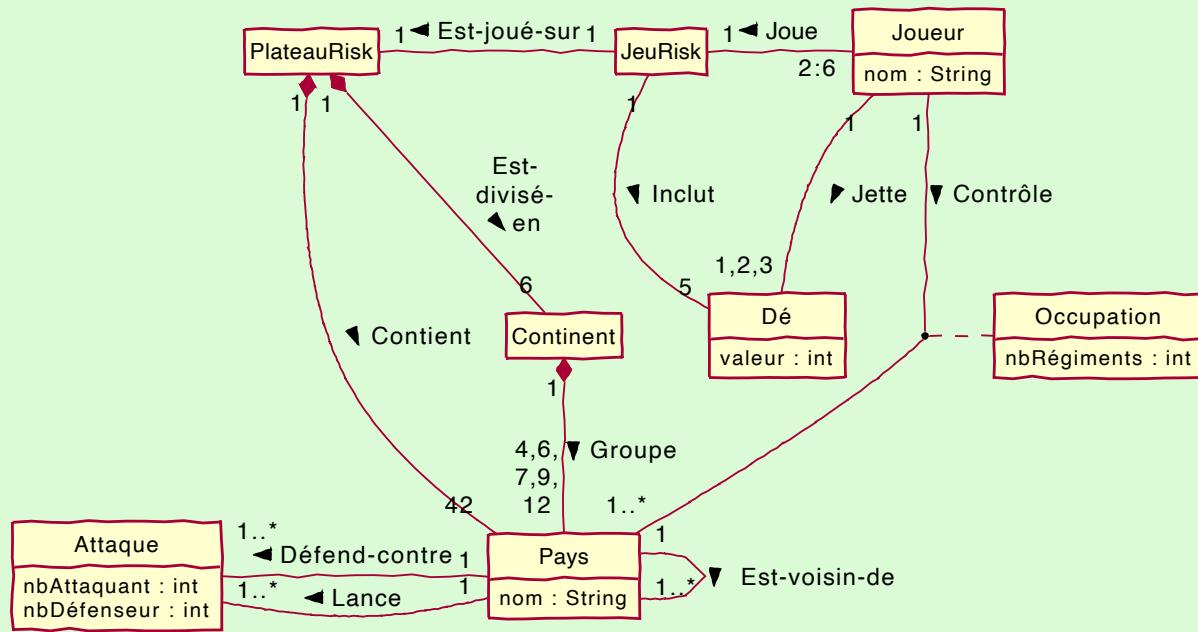


derived attribute from
the multiplicity value

* ATTRIBUT DÉRIVÉ - RISK

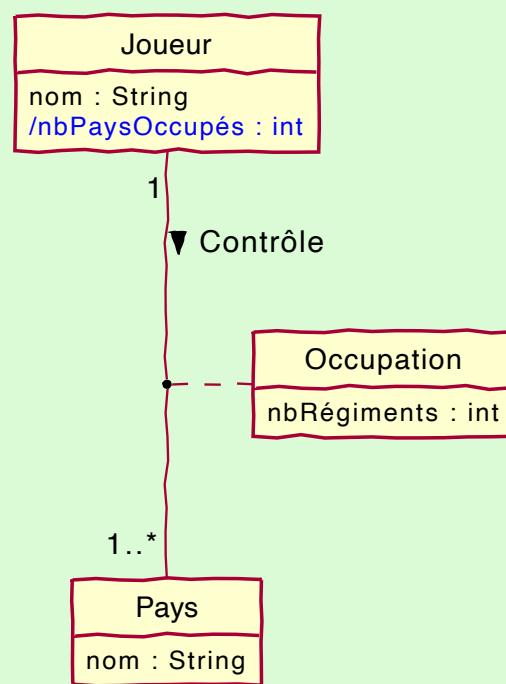
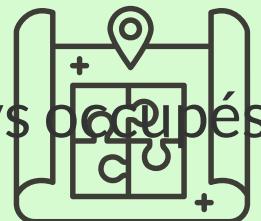


Voyez-vous où appliquer un attribut dérivé?



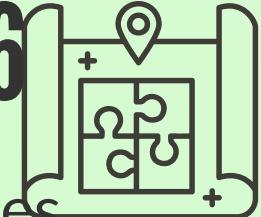
CALCUL DES RENFORTS

On reçoit les renforts selon le nombre de pays occupés.
Utiliser une classe d'association

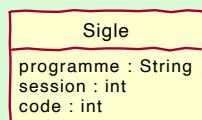


10 . 23

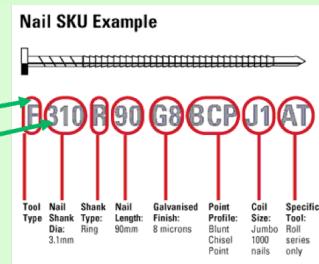
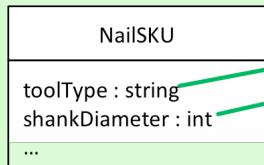
TYPES DE DONNÉES CH. 9.16



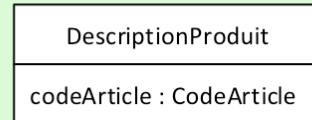
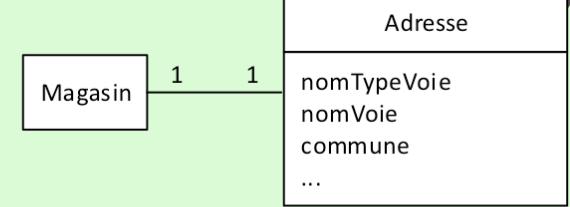
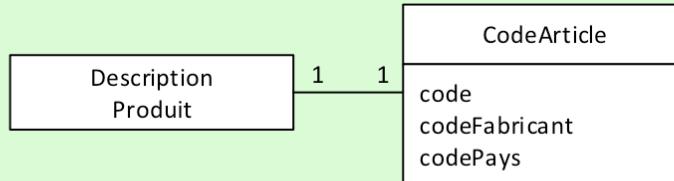
- Définir de nouveaux types quand les types primitifs ne sont pas adéquats
 - ex : Sigle de cours → LOG120



- ex : un code SKU



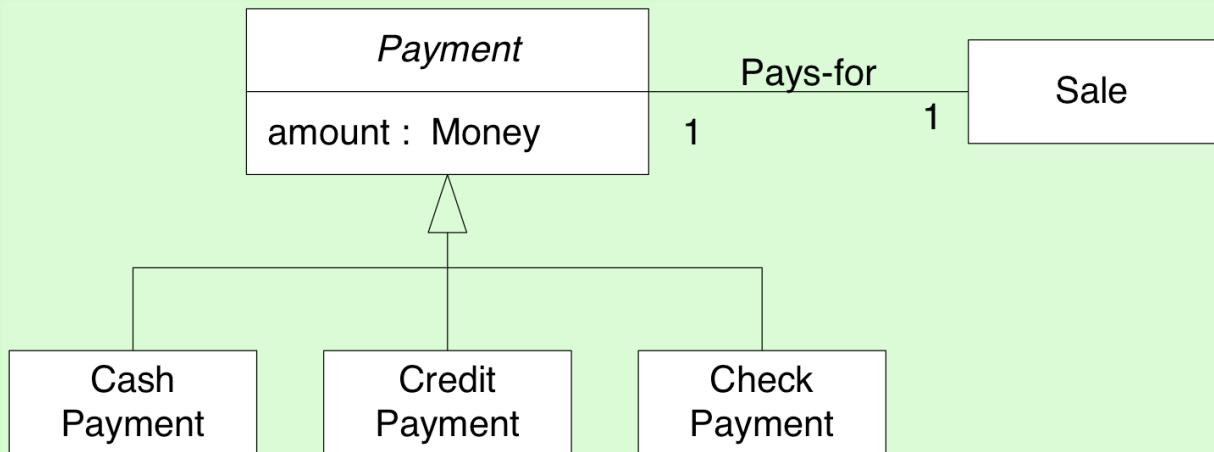
UML : CLASSES TYPES DE DONNÉES



*GÉNÉRALISATION



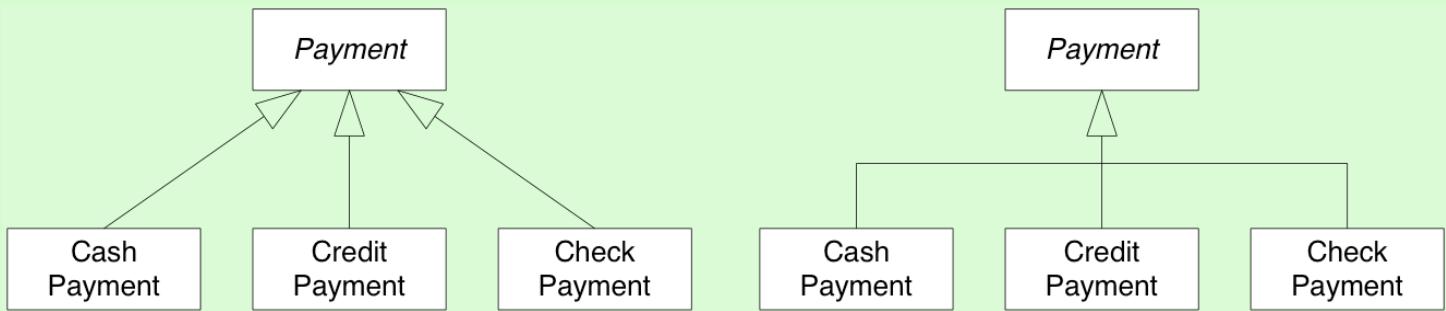
Lire Ch. 32.2-32.7 (version française 26.2-~~26.7~~)



SOUS-CLASSES CONCEPTUELLES



Les énoncés sur les super-classes s'appliquent aux sous-classes



RÈGLE « À 100% »

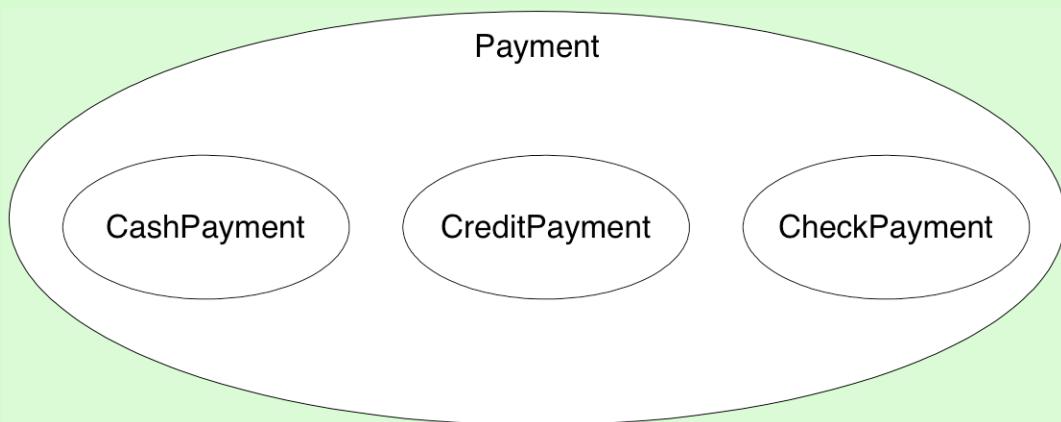


- 100% de la définition d'une classe conceptuelle est applicable aux sous-classes
- Sous-classe
 - conforme à 100% à sa superclasse
 - en termes d'attributs
 - en termes d'associations

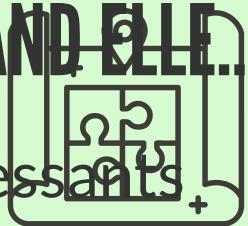
RÈGLE « EST UN »



- Sous-classe conceptuelle
 - est membre de l'ensemble de sa superclasse
 - objet d'une sous-classe **est un** objet de la super-classe

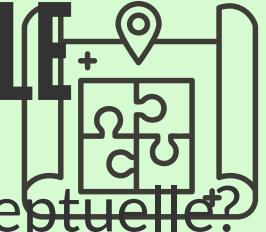


DÉFINIR UNE SOUS-CLASSE CONCEPTUELLE QUAND ELLE...



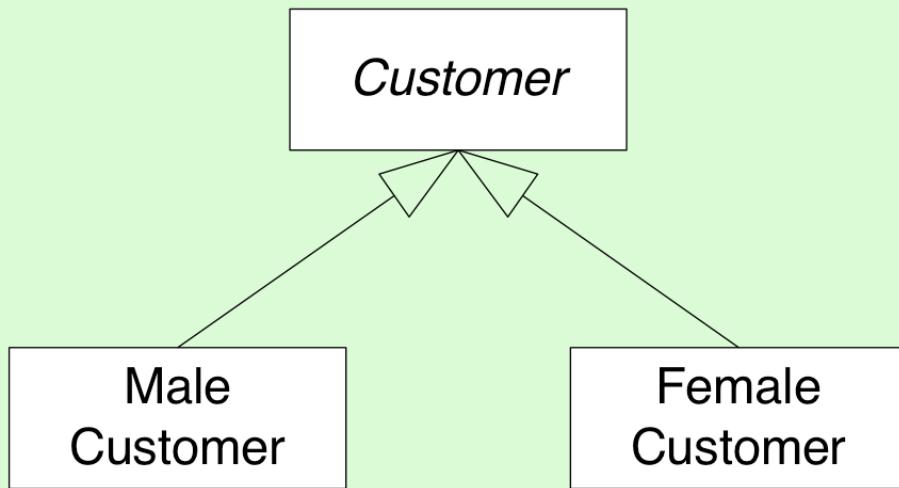
- possède des attributs additionnels intéressants
- possède des associations additionnelles intéressantes
- réagit, est manipulée, est traitée d'une façon intéressante et
 - différemment de sa superclasse
 - différemment d'autres sous-classes
- représente un objet animé (humain, animal, robot) ayant un comportement différent et intéressant

SUPER-CLASSE CONCEPTUELLE



- Quand définit-on une super-classe conceptuelle?
 - Lorsque les classes sont des variations de concepts similaires
 - respectent les règles à 100% et « est un »
 - ont des attributs communs
 - qui pourraient être placés dans la super-classe
 - ont les mêmes associations
 - qui pourraient s'appliquer à la super-classe

EXEMPLE DE PARTITION INUTILE



Correct subclasses.

But useful?

figure: F26.6, A32.6

CLASSES D'ASSOCIATION

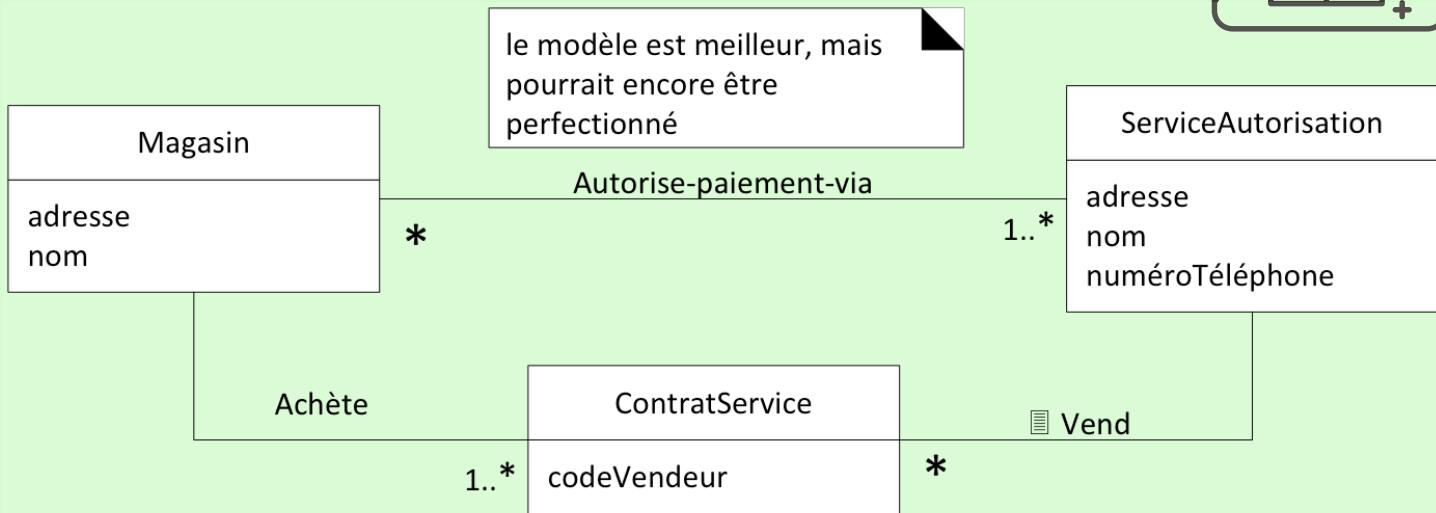
- Services d'autorisation de paiement
 - assignent une identification à chaque magasin
 - demande d'autorisation de paiement ↗ codeVendeur
 - le même magasin peut utiliser plusieurs services, chacun lui donnant un codeVendeur différent
- Dans quelle classe conceptuelle mettre codeVendeur?

| Magasin |
|--------------------|
| adresse |
| codeVendeur |
| nom |

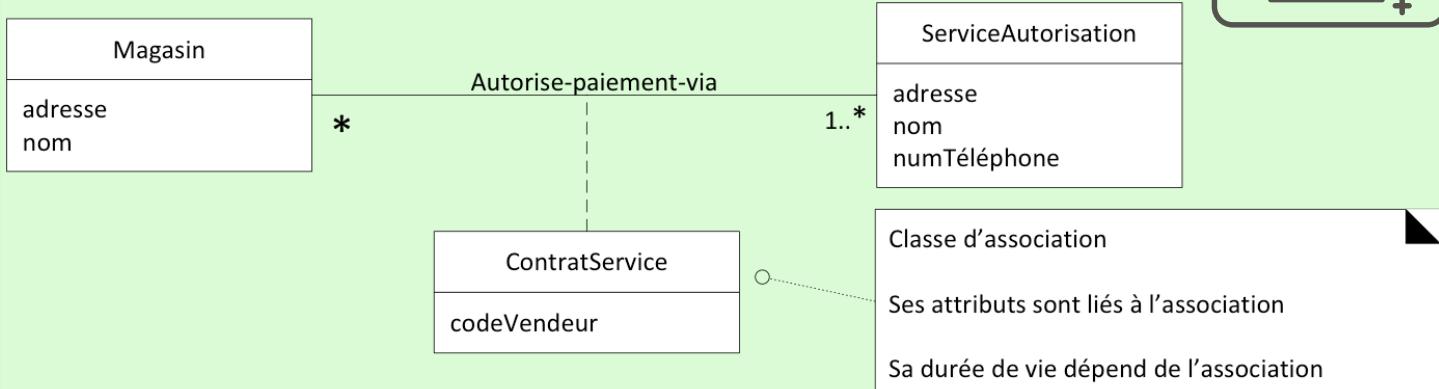
les deux localisations de
codeVendeur sont incorrects,
puisque il peut y en avoir plus d'un

| ServiceAutorisation |
|---------------------|
| adresse |
| codeVendeur |
| nom |
| numTéléphone |

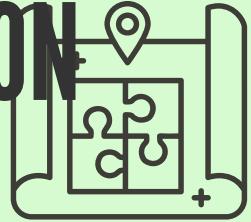
SOLUTION INTERMÉDIAIRE



CLASSE D'ASSOCIATION



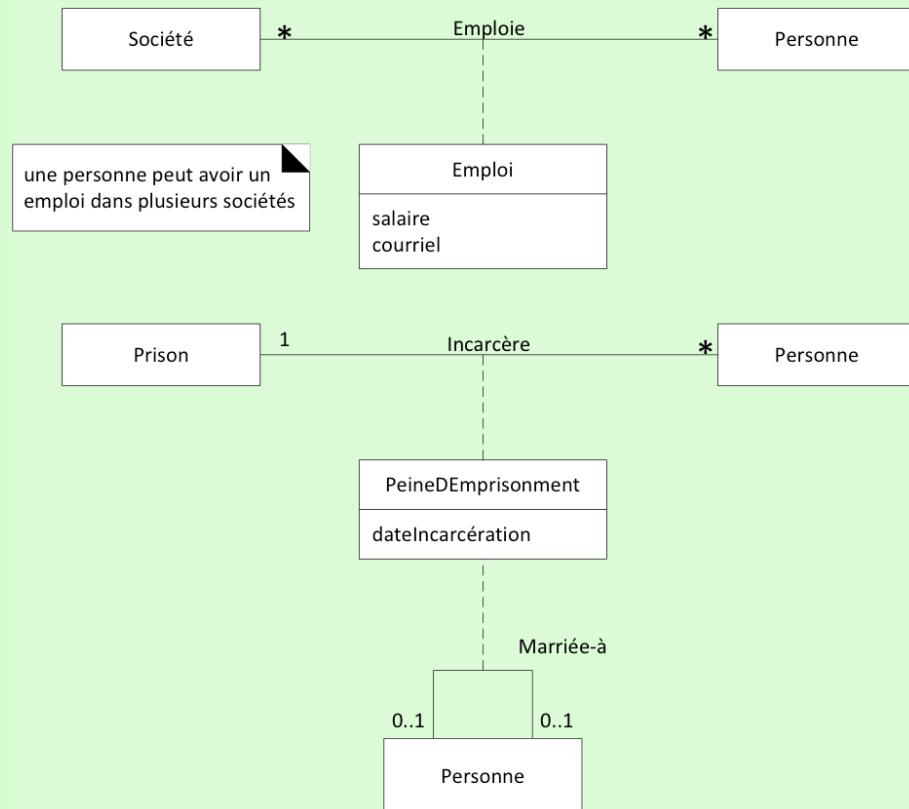
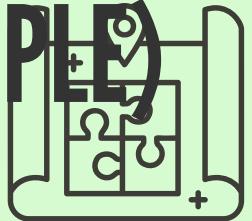
AJOUT - CLASSE D'ASSOCIATION



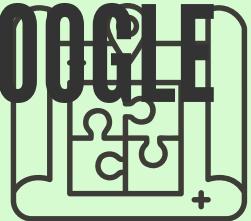
- Règles pour l'ajout :
 - un **attribut** est lié à une association
 - la durée de vie des instances de la classe d'association dépend de l'association
 - «historic mapping» (patron d'analyse, M. Fowler)
- il y a une association N-N entre deux concepts et des **informations liées à l'association** elle-même.

<https://martinfowler.com/apsupp/properties.pdf>

CLASSES D'ASSOCIATION (EXEMPLE)



*SOLUTION EXERCICE MDD SUR GOOGLE CLASSROOM



voir: seance-03-exercice-reserverLivre

EXERCICE MDD (HÔTEL)



1. Un client appelle à l'hôtel pour placer une réservation.
2. Le commis démarre une nouvelle réservation.
3. Le commis saisit la date d'arrivée; la date de départ; le nom de la catégorie de chambre; la quantité de chambres.
4. Le système inscrit les informations à la réservation.

Les étapes 3 et 4 sont répétées tant que le client n'indique pas qu'il a terminé

5. Le commis termine la réservation.
6. Le système affiche toutes les informations entrées.
7. Le commis valide les informations auprès du client et confirme la réservation à l'aide du nom et du numéro de téléphone du client.
8. Le système enregistre la réservation et affiche le numéro de confirmation.
9. Le commis communique le numéro de confirmation au client.

voir seance-01-exercice-CU02

10 . 39

MDD

Correction de l'exercice



10 . 40

MDD?

- Attributs dérivées?



10 . 41

MDD?



- Attributs dérivées?
- Généralisation/spécialisation?

MDD?



- Attributs dérivées?
- Généralisation/spécialisation?
 - Règle « à 100% »?

MDD?



- Attributs dérivées?
- Généralisation/spécialisation?
 - Règle « à 100% »?
 - Règle « est un »?

MDD?



- Attributs dérivées?
- Généralisation/spécialisation?
 - Règle « à 100% »?
 - Règle « est un »?
 - Création d'une sous-classe conceptuelle?

MDD?



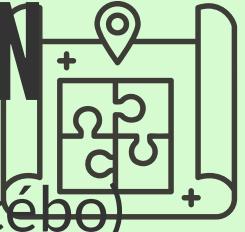
- Attributs dérivées?
- Généralisation/spécialisation?
 - Règle « à 100% »?
 - Règle « est un »?
 - Création d'une sous-classe conceptuelle?
 - Création d'une super-classe conceptuelle?

MDD?

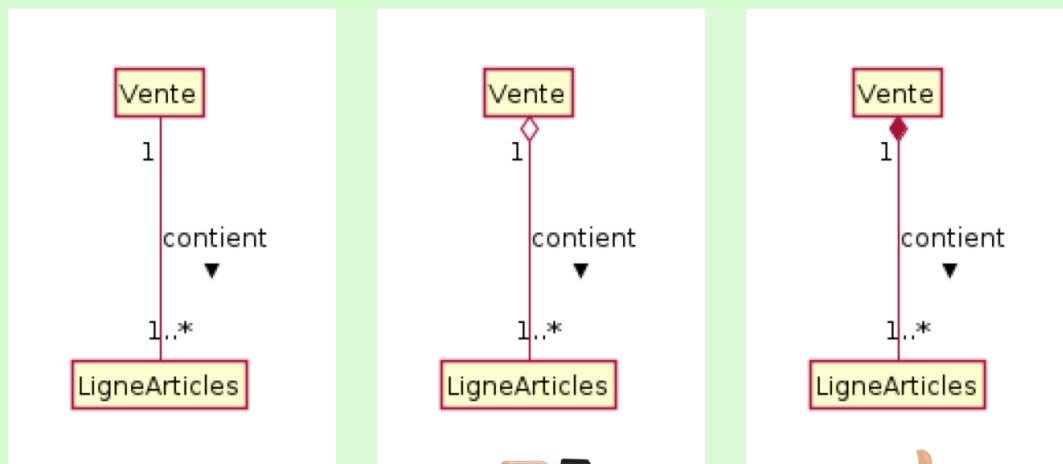


- Attributs dérivées?
- Généralisation/spécialisation?
 - Règle « à 100% »?
 - Règle « est un »?
 - Création d'une sous-classe conceptuelle?
 - Création d'une super-classe conceptuelle?
- Classe d'association

AGRÉGATION ET COMPOSITION



- N'utilisez pas l'agrégation (c'est un placebo)
- Utilisez la composition lorsque c'est approprié.



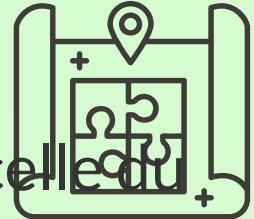
COMPOSITION?



10 . 43

COMPOSITION?

Durée de vie du composant est limitée à celle du composite;



COMPOSITION?



Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.

COMPOSITION?



Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.
- **Assemblage logique ou physique évident** entre le tout et les parties

COMPOSITION?



Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.
- **Assemblage logique ou physique évident** entre le tout et les parties
- **Propriétés** du composite (ex. son emplacement s'étendent aux composants)

COMPOSITION?



Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.
- **Assemblage logique ou physique évident** entre le tout et les parties
- **Propriétés** du composite (ex. son emplacement s'étendent aux composants)
- **Opérations** appliquées au composite (destruction, déplacement et enregistrement) se propagent aux composants

COMPOSITION

AVANTAGES

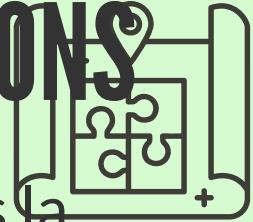


- Clarifie des contraintes du domaine
- Aide à identifier le **patron créateur**
- Opérations sont souvent propagées aux composants

INCONVÉNIENT

- «Chicanerie» - laisser tomber la composition en cas de doute ou de conflit dans l'équipe (voir Bikeshedding)

MODÉLISATION DES ASSOCIATIONS



- La création d'une instance ne signifie pas la création d'une association
- Association un à plusieurs (!dans la solution)
 - Avec un attribut
 - Avec une map (tableau associatif)
 - Avec une liste

CLASSE DE « DESCRIPTION »

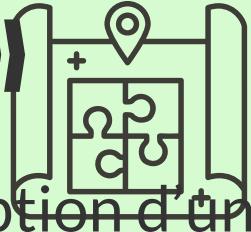


| Article |
|---|
| description prix numéro de série codeArticle |

Déconseillé

- Article représente un article physique
- Attributs prix, numéro de série et code
- Chaque fois qu'un article est vendu, l'objet Article correspondant est « supprimé »

CLASSE DE « DESCRIPTION »



- Il est nécessaire de disposer de la description d'un produit ou d'un service, indépendamment de l'existence actuelle de ces derniers
- La suppression d'instances d'une entité qu'elle décrit entraîne la perte d'information qui doit être mémorisée
- Elle réduit la redondance ou la duplication des informations

CLASSES DE DESCRIPTION

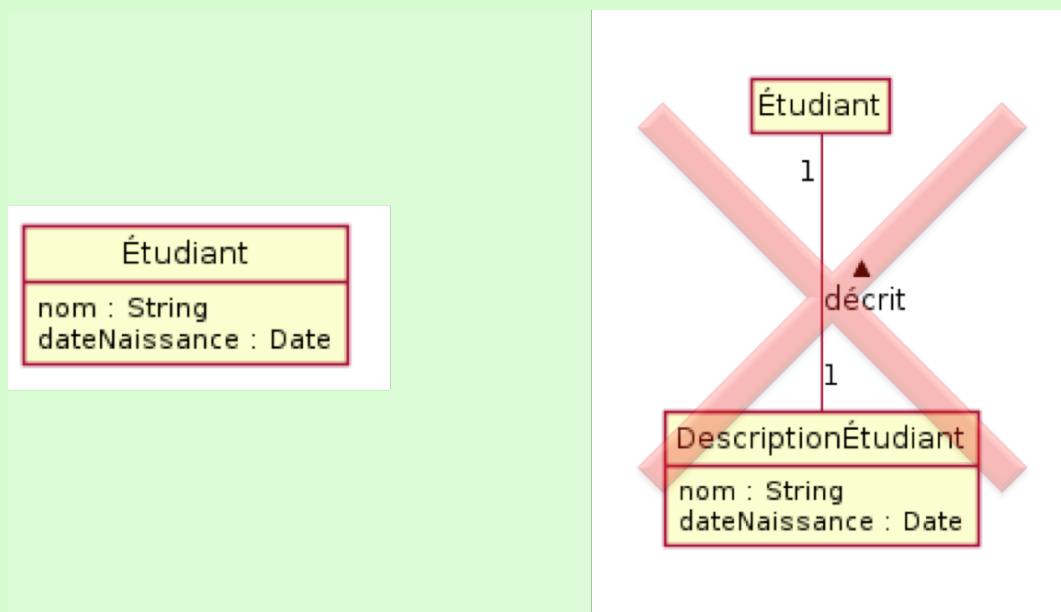


- Souvent, les objets d'une classe de description sont stockés dans un Catalogue
- Lire en détail la section 9.13 du livre pour d'autres exemples

CLASSES DE DESCRIPTION



- Attention : ne pas appliquer les classes de description n'importe comment:

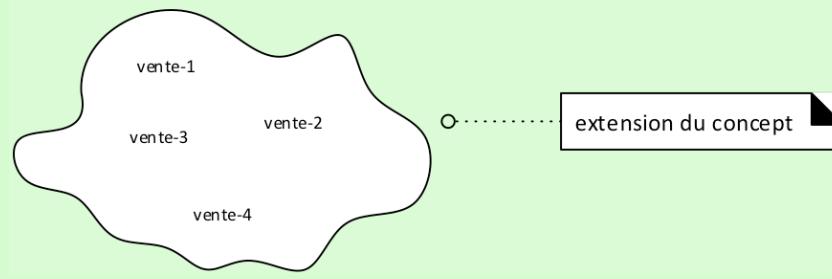
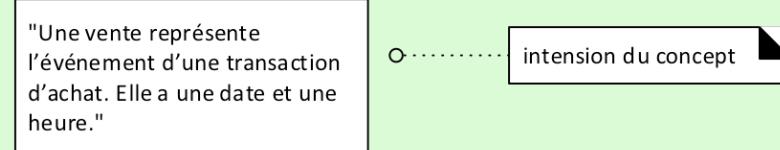
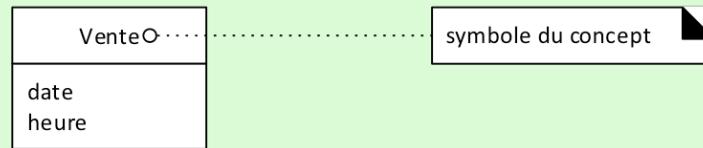
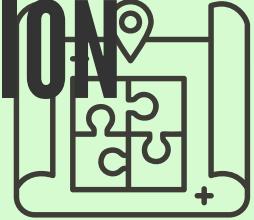


MDD?

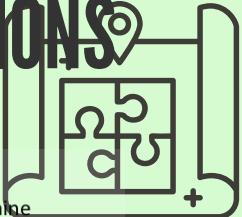


- Agrégation vs composition?
- Comment interpréter une multiplicité de plusieurs

SYMBOLE, INTENSION, EXTENSION



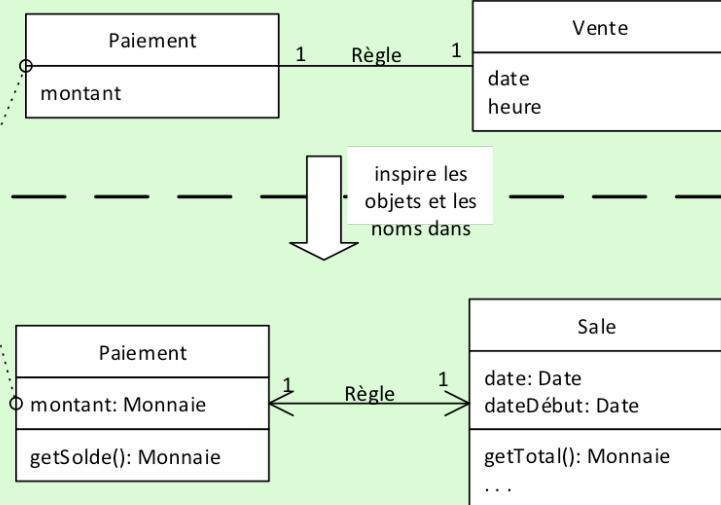
RÉDUIRE LE DÉCALAGE DES PRÉSENTATIONS



Dans le Modèle du Domaine, un Paiement est un concept, mais c'est une classe logicielle dans le Modèle de Conception. Les deux ne sont pas identiques, et le premier a *inspiré* le nom et la définition de la seconde.

Cela réduit le décalage des représentations, et constitue l'une des grandes idées de la technologie objet.

Modèle du Domaine du Processus Unifié
Façon dont les parties prenantes voient le concepts significatifs du domaine



Modèle de Conception du Processus Unifié
Le développeur orienté objet s'est inspiré du domaine du monde réel pour créer les classes logicielles.

En conséquence, le décalage des représentations entre la façon dont les parties prenantes voient le domaine et sa traduction logicielle a été réduit.

CHAP 9

1. Qu'est-ce qu'un modèle du domaine?



CHAP 9



1. Qu'est-ce qu'un modèle du domaine?
2. Que veut dire symbole, intension et extension quand on parle d'un MDD?

CHAP 9



1. Qu'est-ce qu'un modèle du domaine?
2. Que veut dire symbole, intension et extension quand on parle d'un MDD?
3. Qu'est-ce que le décalage des représentations?

CHAP 9



1. Qu'est-ce qu'un modèle du domaine?
2. Que veut dire symbole, intension et extension quand on parle d'un MDD?
3. Qu'est-ce que le décalage des représentations?
4. Qu'est-ce qu'une classe de description?

CHAP 9



1. Qu'est-ce qu'un modèle du domaine?
2. Que veut dire symbole, intension et extension quand on parle d'un MDD?
3. Qu'est-ce que le décalage des représentations?
4. Qu'est-ce qu'une classe de description?
5. Quand doit-on représenter une association?

CHAP 9



1. Qu'est-ce qu'un modèle du domaine?
2. Que veut dire symbole, intension et extension quand on parle d'un MDD?
3. Qu'est-ce que le décalage des représentations?
4. Qu'est-ce qu'une classe de description?
5. Quand doit-on représenter une association?
6. Décrivez la notation UML d'un MDD

CHAP 9



1. Qu'est-ce qu'un modèle du domaine?
2. Que veut dire symbole, intension et extension quand on parle d'un MDD?
3. Qu'est-ce que le décalage des représentations?
4. Qu'est-ce qu'une classe de description?
5. Quand doit-on représenter une association?
6. Décrivez la notation UML d'un MDD
7. Qu'est-ce qu'un exemple des associations multiples?

CHAP 9



1. Qu'est-ce qu'un modèle du domaine?
2. Que veut dire symbole, intension et extension quand on parle d'un MDD?
3. Qu'est-ce que le décalage des représentations?
4. Qu'est-ce qu'une classe de description?
5. Quand doit-on représenter une association?
6. Décrivez la notation UML d'un MDD
7. Qu'est-ce qu'un exemple des associations multiples?
8. Qu'est-ce qu'un attribut dérivé? Notation ?

CHP9 COMPLEXITÉ

1. Qu'est-ce que la complexité inhérente?



10 . 54

CHP9 COMPLEXITÉ

1. Qu'est-ce que la complexité inhérente?
2. Donnez un synonyme de inhérente.



CHP9 COMPLEXITÉ



1. Qu'est-ce que la complexité inhérente?
2. Donnez un synonyme de inhérente.
3. Qu'est-ce que la complexité accidentelle?

CHP9 COMPLEXITÉ



1. Qu'est-ce que la complexité inhérente?
2. Donnez un synonyme de inhérente.
3. Qu'est-ce que la complexité accidentelle?
4. La complexité inhérente est-elle due au problème ou à la solution?

CHP9 COMPLEXITÉ



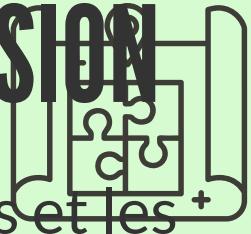
1. Qu'est-ce que la complexité inhérente?
2. Donnez un synonyme de inhérente.
3. Qu'est-ce que la complexité accidentelle?
4. La complexité inhérente est-elle due au problème ou à la solution?
5. Est-il plus facile de gérer la complexité inhérente ou accidentelle? Pourquoi?

CHP9 COMPLEXITÉ



1. Qu'est-ce que la complexité inhérente?
2. Donnez un synonyme de inhérente.
3. Qu'est-ce que la complexité accidentelle?
4. La complexité inhérente est-elle due au problème ou à la solution?
5. Est-il plus facile de gérer la complexité inhérente ou accidentelle? Pourquoi?
6. Comment peut-on gérer la complexité inhérente?

VALIDATION DE LA COMPRÉHENSION



En UML on modélise les classes conceptuelles et les classes logicielles dans la même perspective.

- Vrai
- Faux

CHP26 AFFINEMENT DU MDD



1. Qu'est-ce que la généralisation?

10 . 56

CHP26 AFFINEMENT DU MDD



1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?

CHP26 AFFINEMENT DU MDD



1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?

CHP26 AFFINEMENT DU MDD



1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?
4. Est-il préférable d'utiliser une agrégation ou une composition? Pourquoi?

CHP26 AFFINEMENT DU MDD



1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?
4. Est-il préférable d'utiliser une agrégation ou une composition? Pourquoi?
5. Qu'est-ce qu'un attribut dérivé?

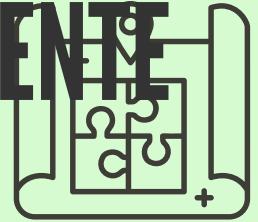
AFFINEMENT DU MODÈLE DU DOMAINE



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

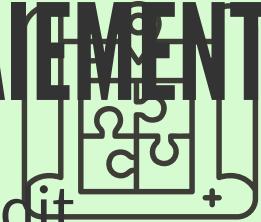
Tableau F26.1/A32.1

EXTENSIONS AU UC1: TRAITER VENTE



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

SYSTÈMES D'AUTORISATION DE PAIEMENT



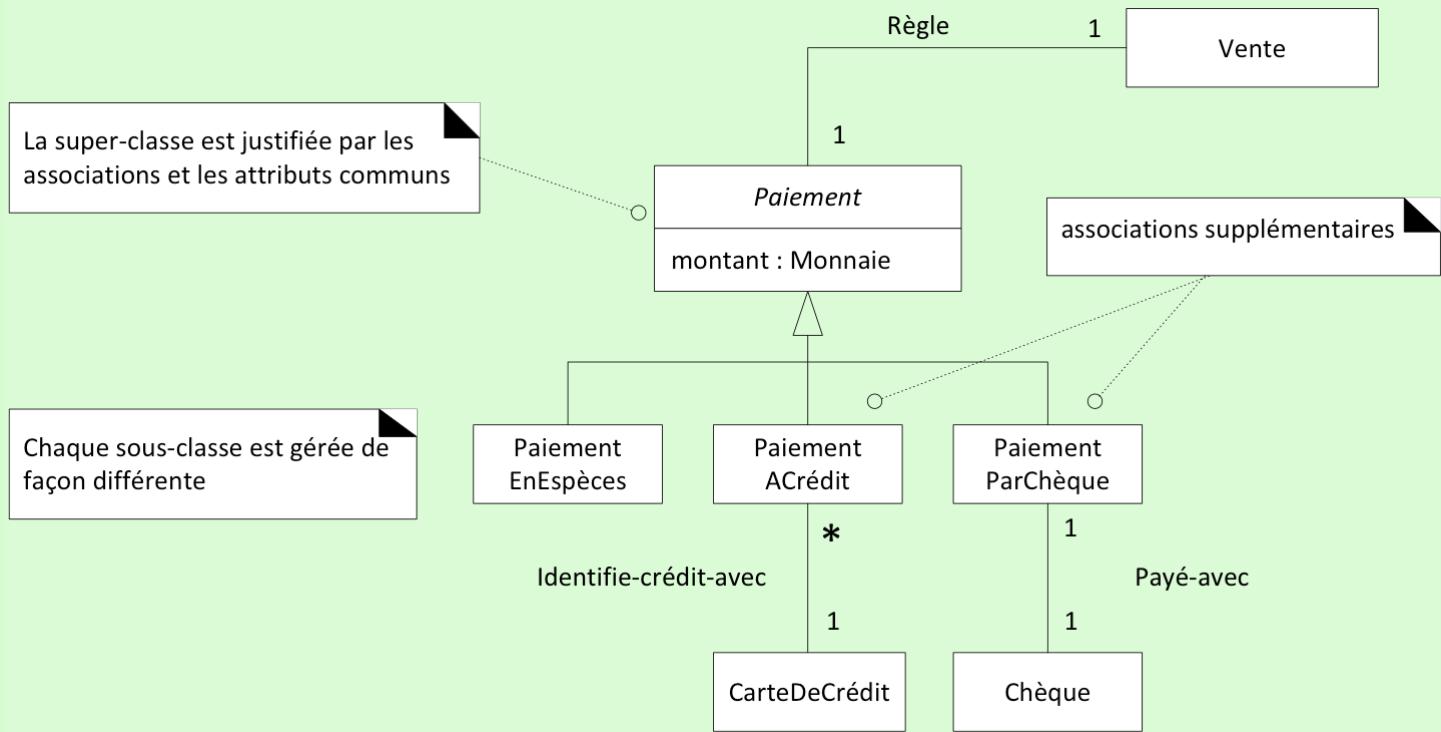
- RequêtePaiementACrédit et AccordCrédit
 - Abstractions des éléments dans l'activité d'autorisation de paiement
 - Pas nécessairement des informations (en termes de données transactionnelles) transmises sur le fil du réseau
 - **Classes conceptuelles toujours**



EXEMPLE DANS POS

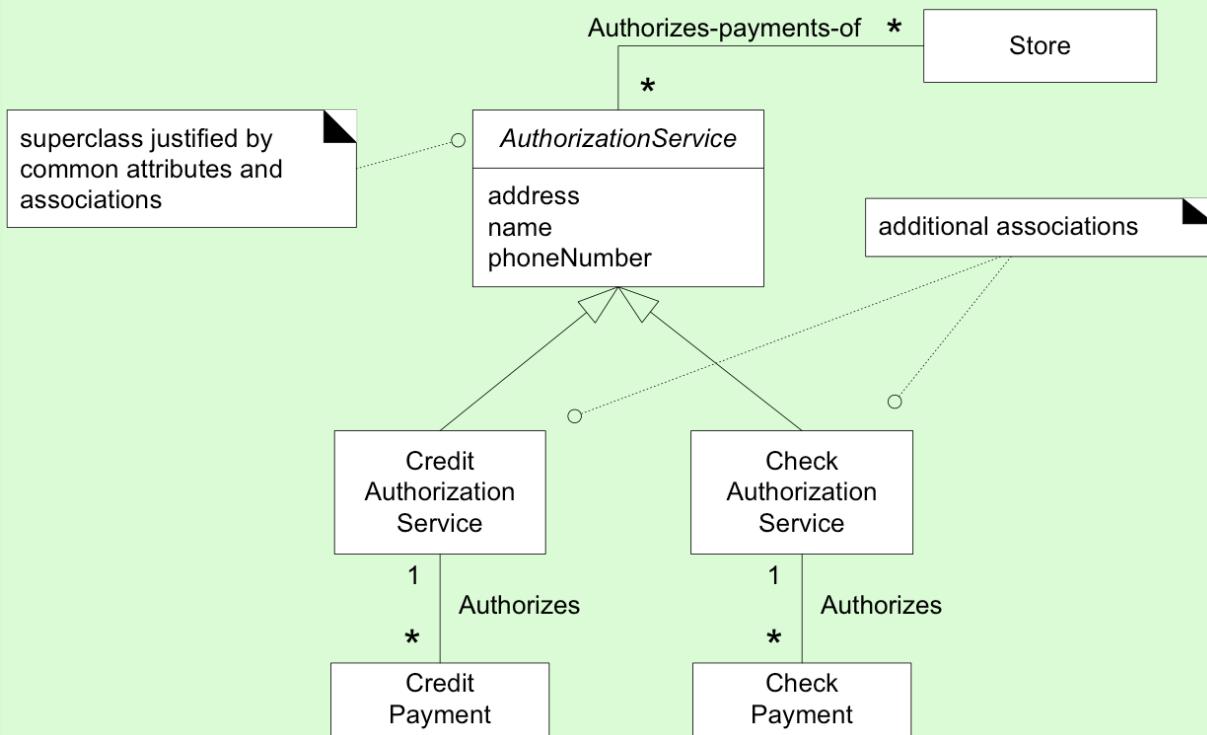


Généralisation/spécialisation



EXEMPLE DANS POS

Hiérarchie des services d'autorisation



TRANSACTIONS D'AUTORISATION

- Est-ce utile?

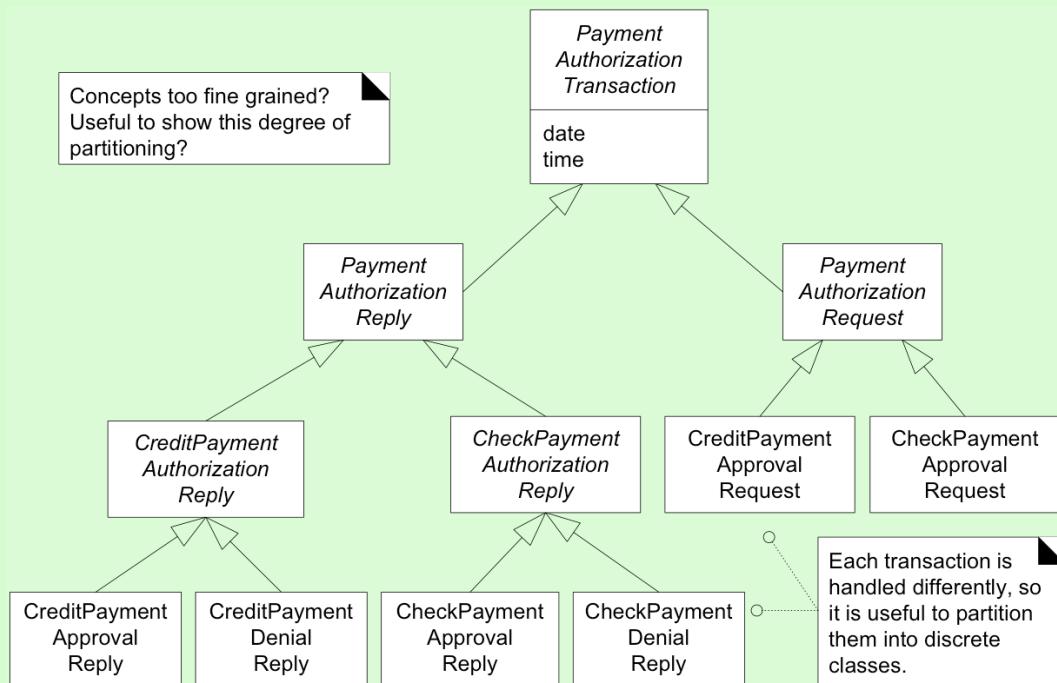
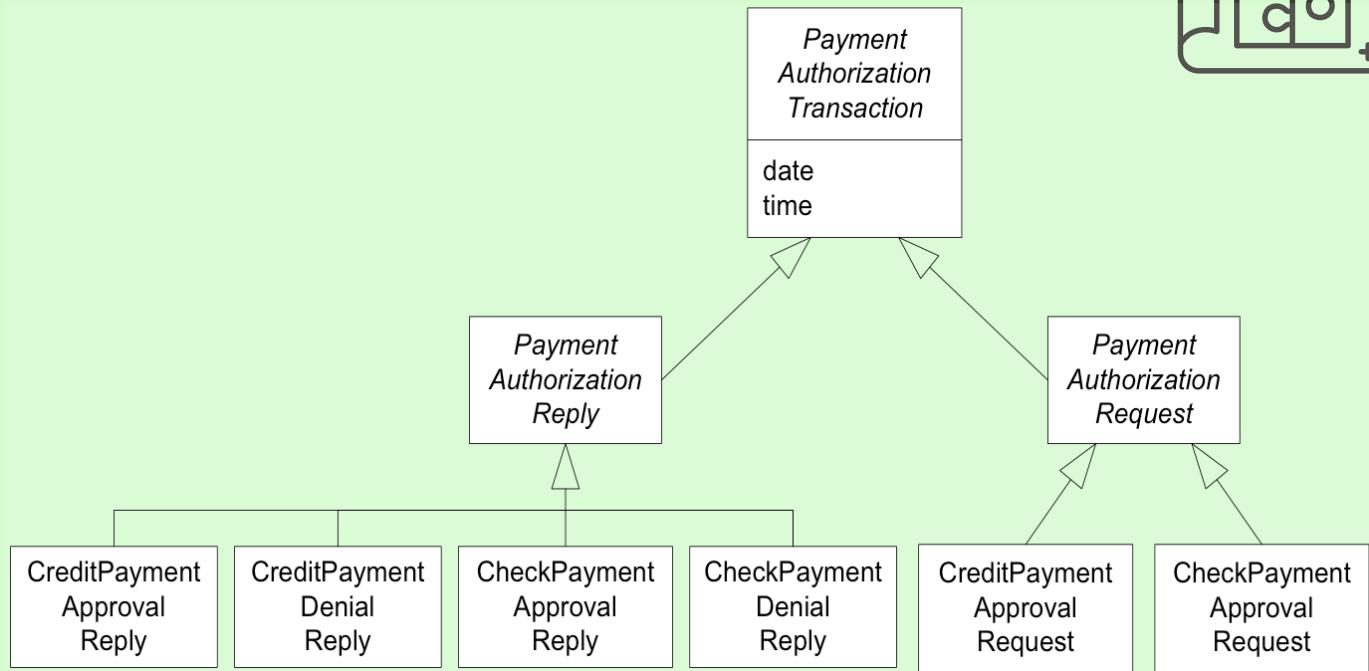


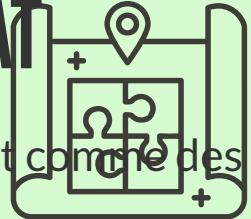
figure: F26.9, A32.9

TRANSACTIONS D'AUTORISATION

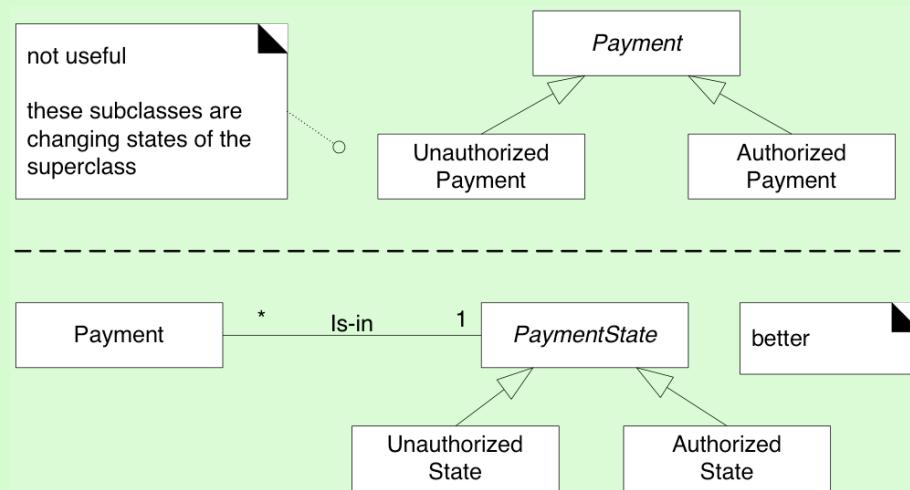


Le modèle du domaine n'est pas jugé en fonction de sa précision (justesse, correcte ou non) mais en terme de son utilité. Il n'est pas un but en soi

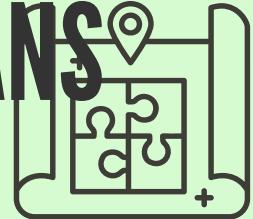
MODÉLISATION DE CHANGEMENT D'ÉTAT



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

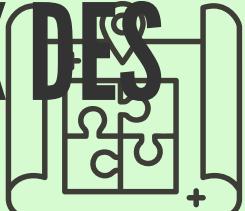


HIÉRARCHIES ET HÉRITAGE DANS L'IMPLÉMENTATION

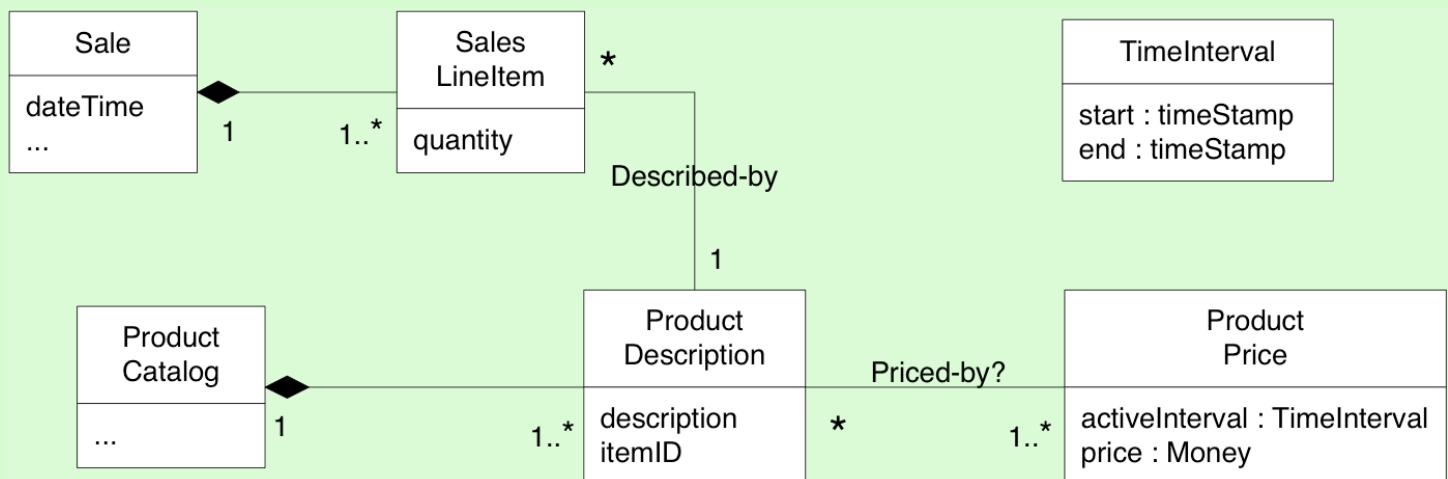


- Les hiérarchies de classes conceptuelles peuvent être reflétées ou non dans le modèle de conception
 - Par exemple, les classes paramétrées (templates) en C++/Java/C# permettent de réduire le nombre de classes dans l'implémentation
 - www.tutorialspoint.com/cplusplus/cpp_templates

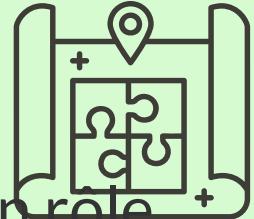
INTERVALLES DE TEMPS ET PRIX DES PRODUITS



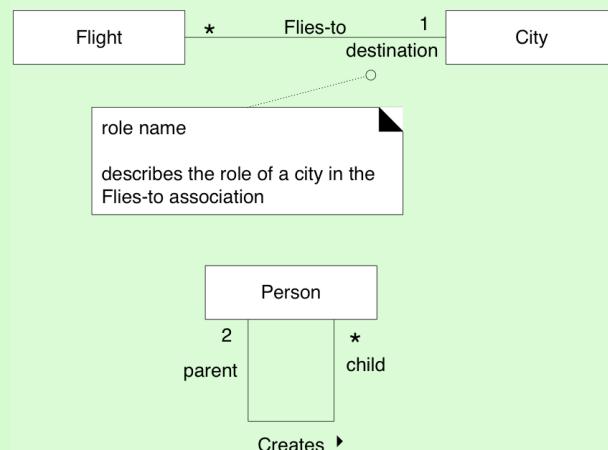
Correction d'une « erreur » de l'itération 1



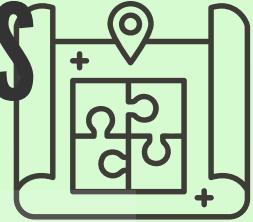
NOM DE RÔLES



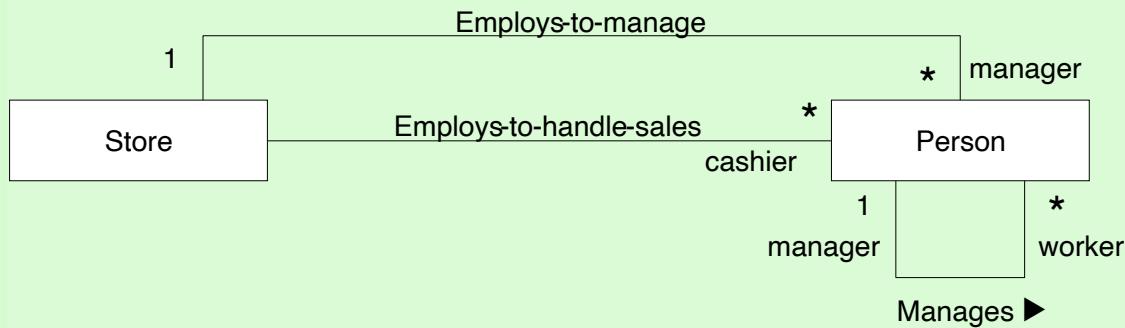
- Extrémité d'une association peut avoir un rôle
- Rôle possède plusieurs propriétés: nom et multiplicité



CONCEPTS VS ASSOCIATIONS



roles in associations



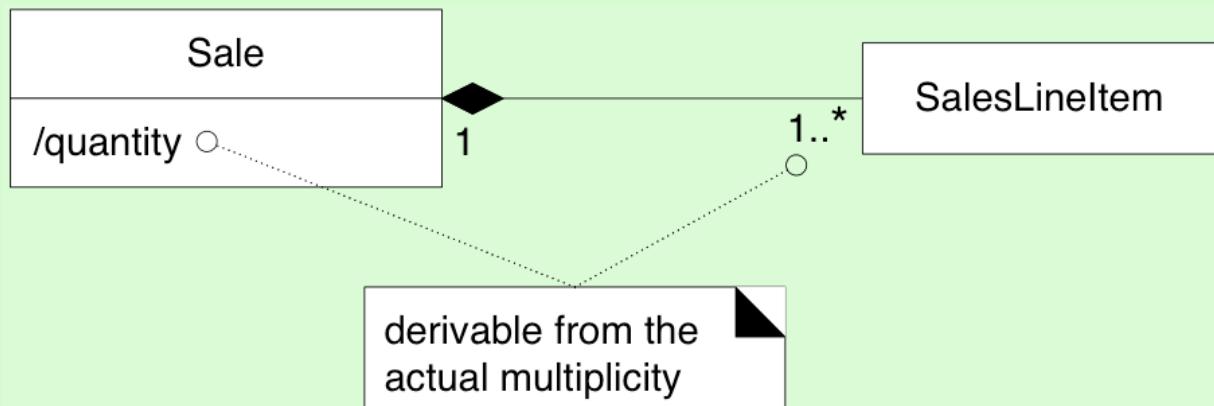
roles as concepts



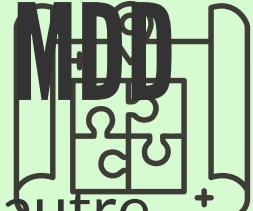
ÉLÉMENTS DÉRIVÉS



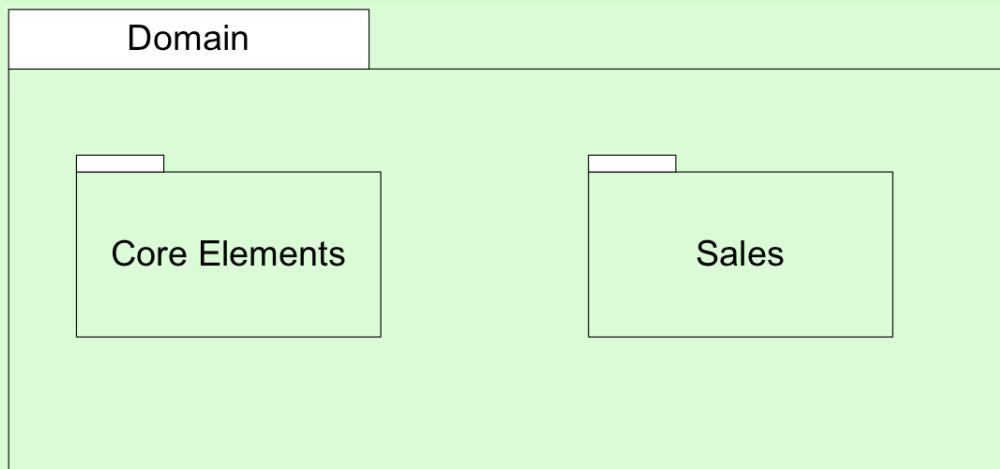
Évitez de faire apparaître les éléments dérivés dans les ~~diagrammes~~,
sauf si leur omission peut nuire à la compréhension



PACKAGES POUR ORGANISER LE MDD



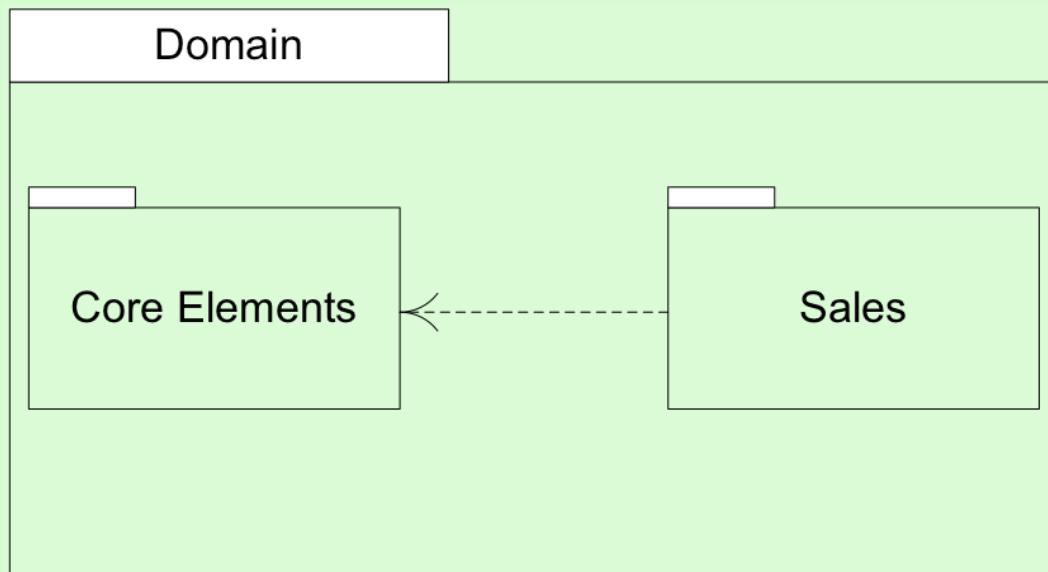
- Un élément peut être référencé dans un autre package
 - nom augmenté du nom du package



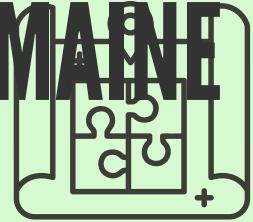
DÉPENDANCE ENTRE PACKAGE



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

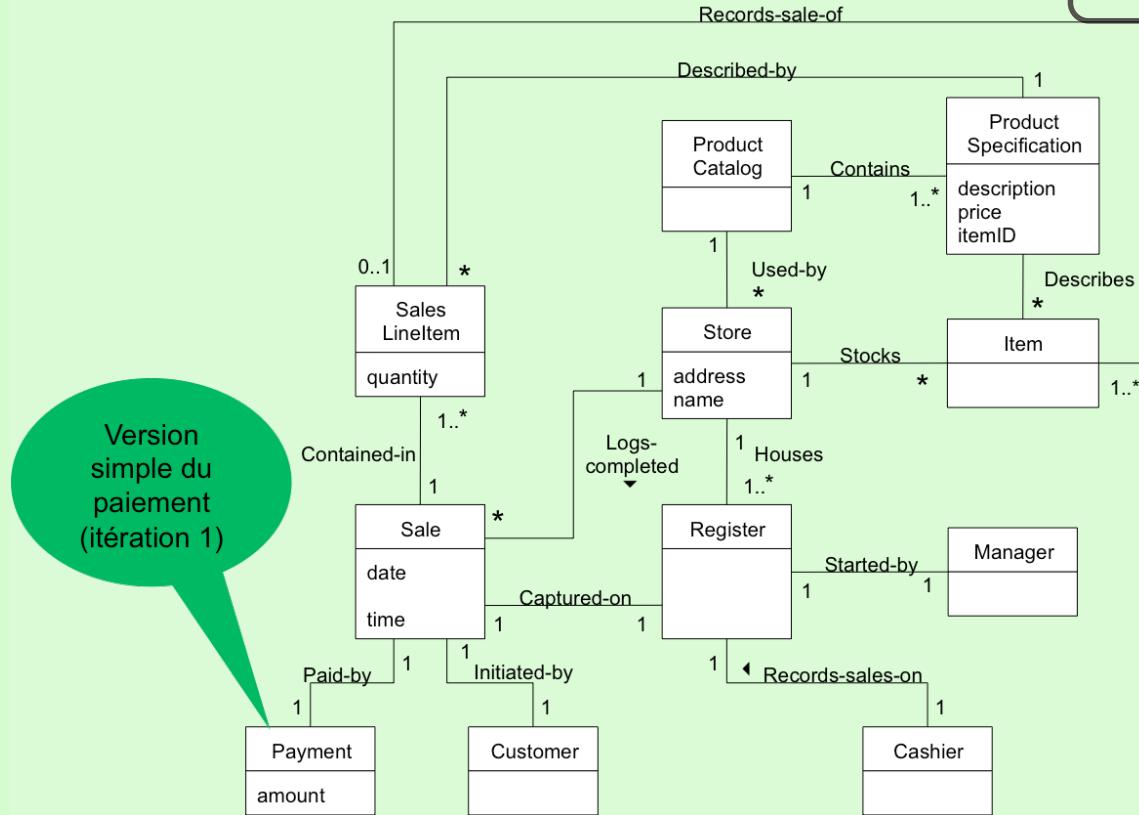


PARTITIONNER LE MODÈLE DU DOMAINE

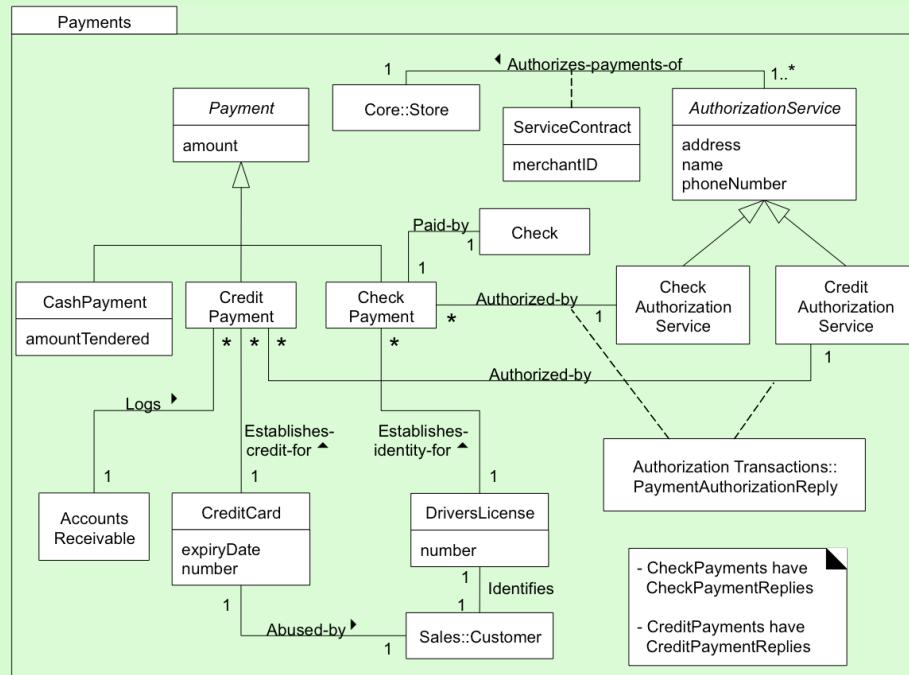


- Placer en package les éléments
 - Cohésion relationnelle
 - qui sont fortement associés
 - situés dans la même hiérarchie de classes
 - Cohésion fonctionnelle
 - situés dans le même domaine
 - participant au même cas d'utilisation

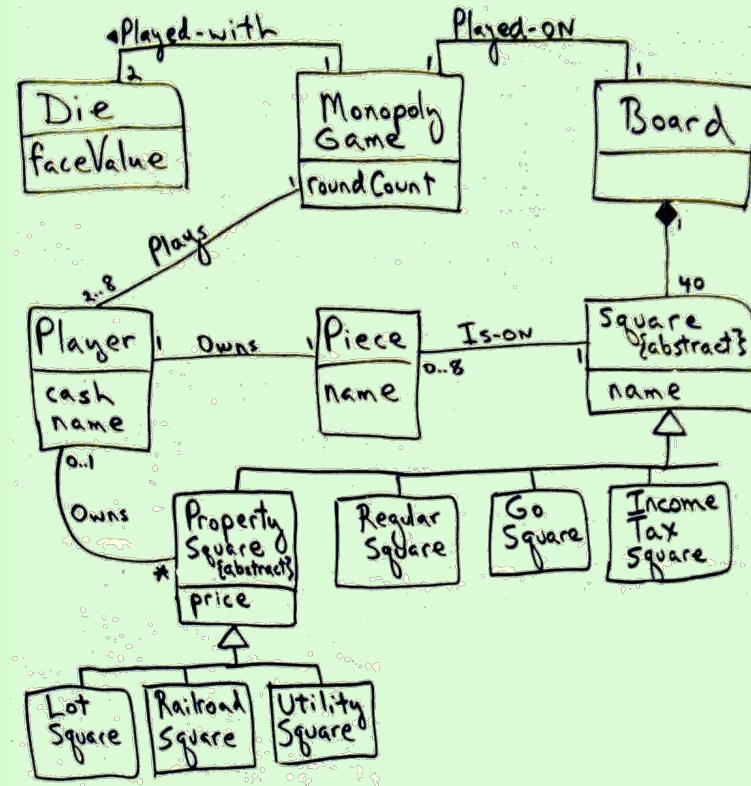
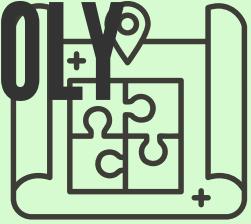
MODÈLE DU DOMAINE (1)



MODÈLE DU DOMAINE (2)



AFFINEMENTS DU MDD MONOPOLY

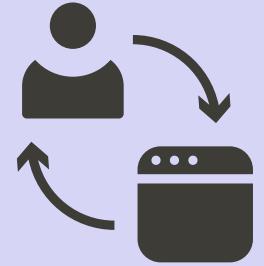


MODULE DSS

1. Qu'est-ce qu'un DSS? - 8.47m
2. Opérations systèmes - 19.01m
3. Exercice - 11.23m
4. Retour d'opération - 4.45m
5. Révision - Diagramme de séquence système - 18.28m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexité](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

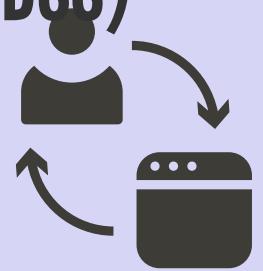
QU'EST-CE QU'UN DSS?



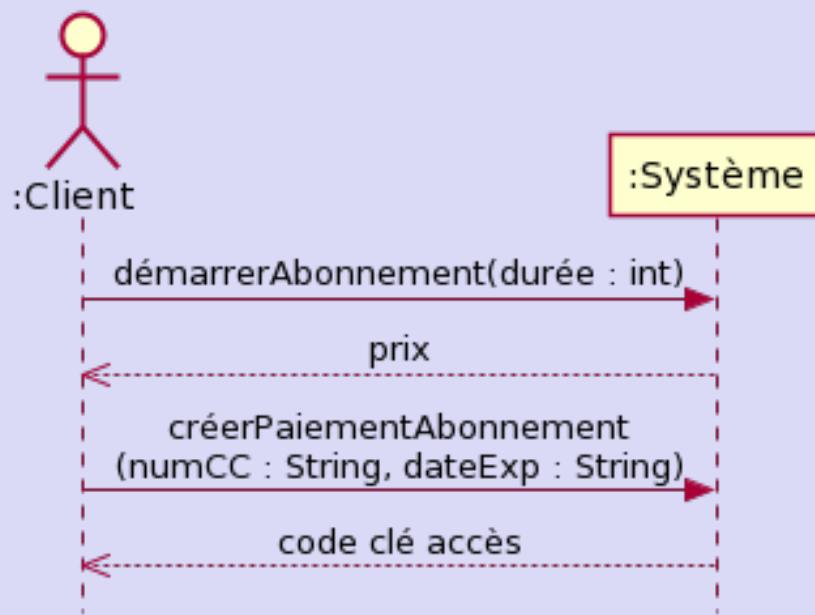
- Interaction avec le système
 - acteur génère des événements
 - réponses du système
- Illustrer ces opérations
 - pour mieux comprendre le système

DIAGRAMME DE SÉQUENCE DU SYSTÈME (DSS)

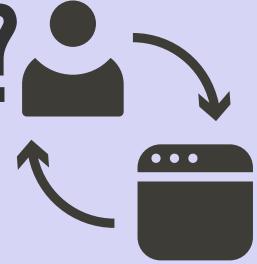
- événements synchrones
- paramètres
- retour d'information
- frontières du système



Traiter abonnement

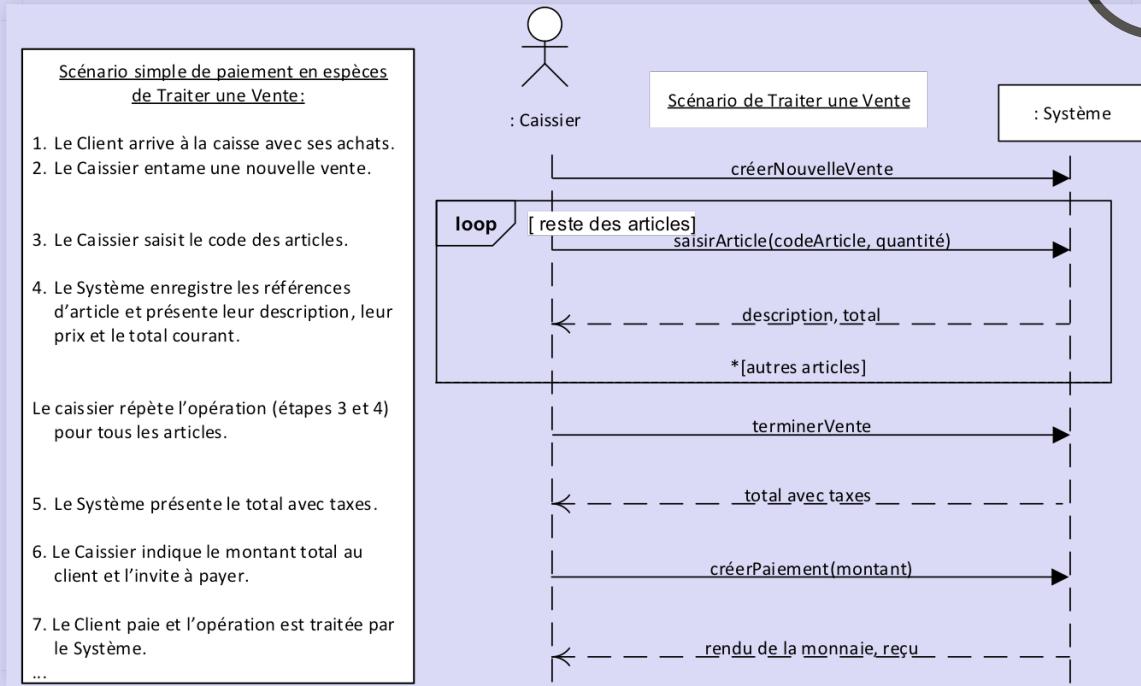
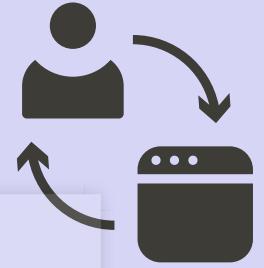


POURQUOI TRACER UN DSS?



- Pour connaître les événements système
 1. déclenchés par acteurs (humains ou ordis)
 2. d'échéance de temporisation (timeout)
 3. d'erreurs ou d'exceptions (sources externes)
- Début de la modélisation par décomposition (top-down)
- Vu abstrait du système comme application
- Facilite le changement de l'interface humain-machine (la couche présentation)

DSS ET CAS D'UTILISATION



<https://plantuml.com/sequence-diagram>

RÉALISER VOTRE PREMIER DSS

- Réaliser un diagramme de séquence système du cas d'utilisation Noter une réservation.



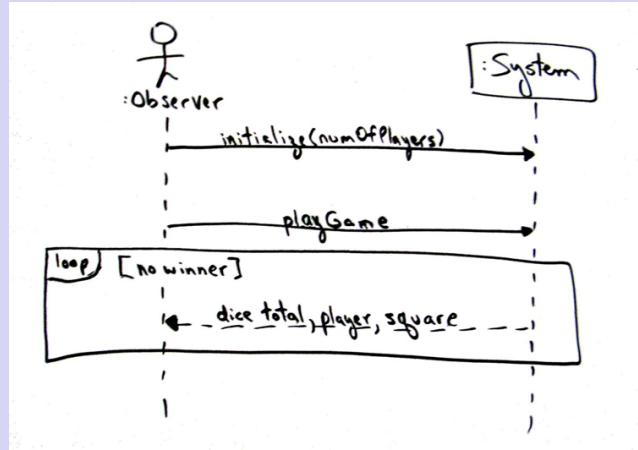
11 . 6

IU - INTERFACE USAGER VS DSS

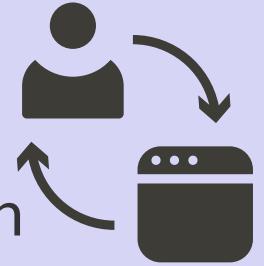
Le DSS peut vous permettre de définir les interfaces usager nécessaires pour appeler les opérations systèmes et afficher les retours d'informations.

- Vérifier la correspondance entre vos interfaces usager et vos opérations systèmes

DIAGRAMME SÉQUENCE SYSTÈME

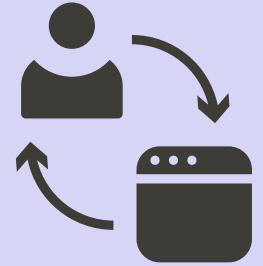


DSS



- Crée à partir d'un cas d'utilisation
- Modélise l'interaction
 - Opérations système
 - Messages de retour (au besoin)
- Notation UML
- But : définir l'API (haut niveau)
- Système est une « boîte noire »

FAIRE UN DSS



- Identifier l'acteur principal
- Modéliser système comme boîte noire
- Proposer une opération système pour chaque évènement système
 - Types primitifs pour arguments
 - Messages de retour si nécessaire

DSS: CU01 - AJOUTER UN LIVRE À ÉCHANGER

- 
1. Le Client démarre un nouvel ajout de livre
 2. Le Client entre le code ISBN du livre, ainsi que le code de sa condition.
 3. Le Système enregistre le livre et présente sa description.

Le Client répète les étapes 2 à 3 jusqu'à ce qu'il ait saisi tous les livres à échanger.

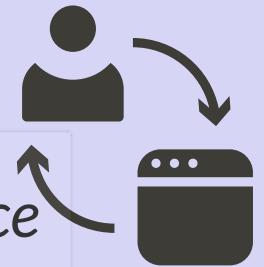
4. Le Système présente la liste de livres que possède le Client.

DSS: «ATTAQUER UN PAYS»



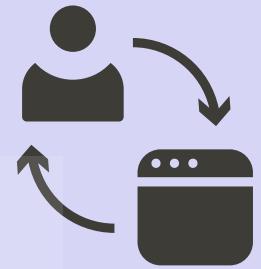
1. Le Joueur attaquant choisit d'attaquer un pays voisin du Joueur défenseur.
2. Le Joueur attaquant annonce combien de régiments il va utiliser pour son attaque.
3. Le Joueur défenseur annonce combien de régiments il va utiliser pour sa défense.
4. Les deux Joueurs jettent le nombre de dés selon leur stratégie choisie aux étapes précédentes.
5. Le Système compare les dés et élimine les régiments de l'attaquant ou du défenseur selon les règles et affiche le résultat.

DSS: «ATTAQUER UN PAYS»

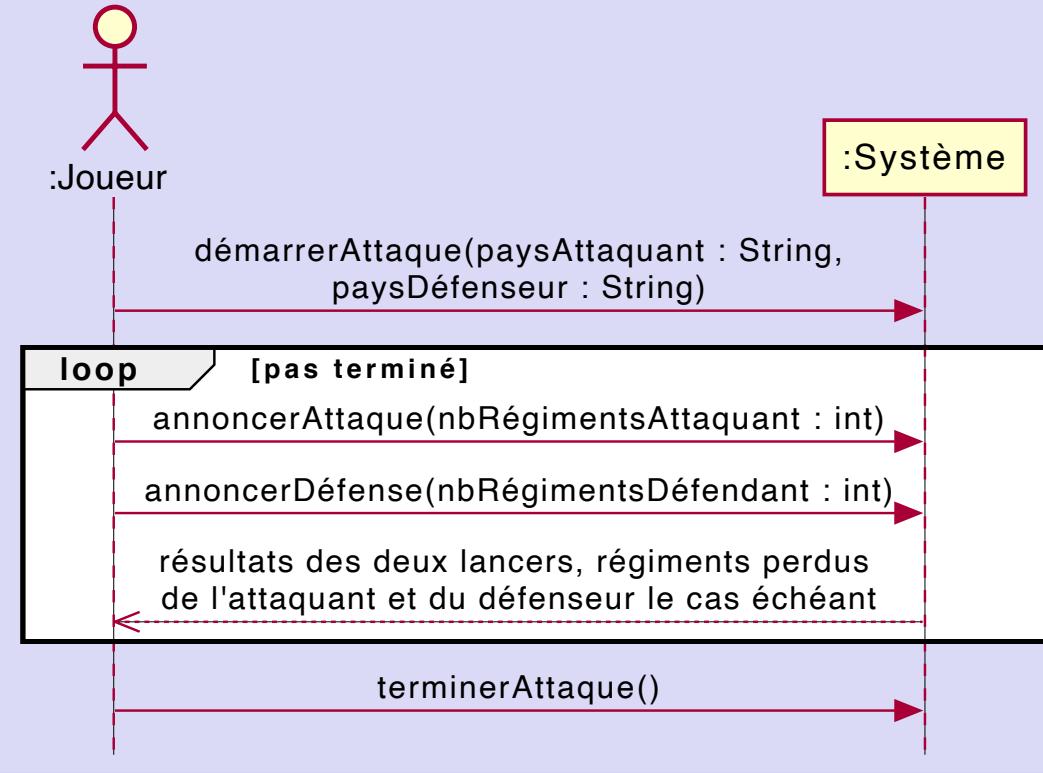


Les Joueurs répètent l'étape 2 jusqu'à ce que l'attaquant ne puisse plus attaquer ou ne veuille plus attaquer.

DSS: «ATTAQUER UN PAYS»

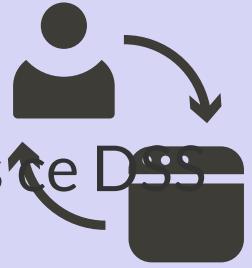


DSS pour *Attaquer un pays*

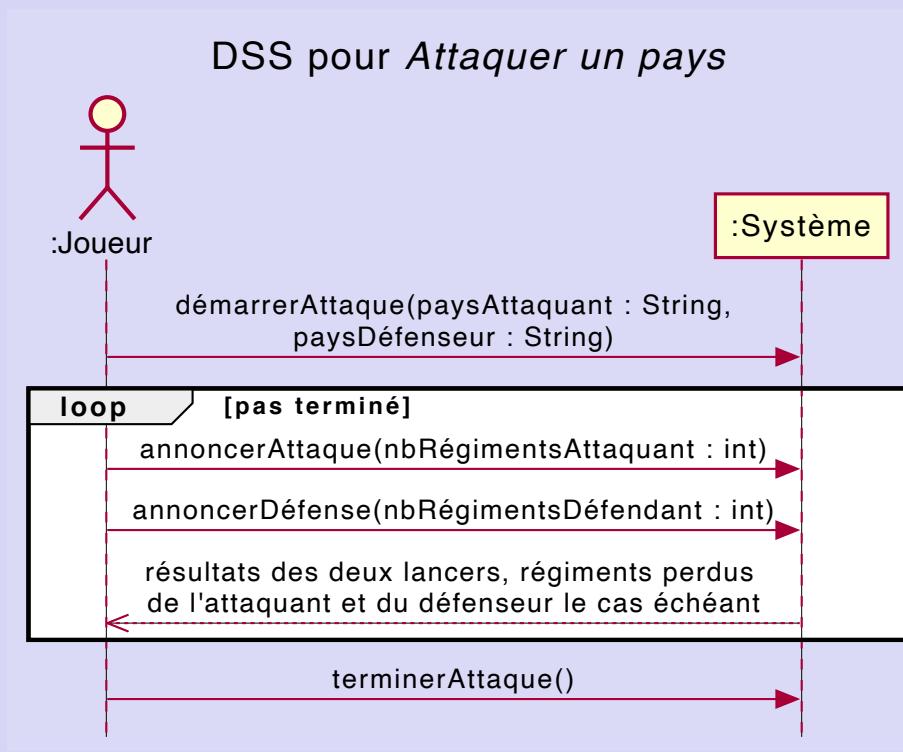


11 . 14

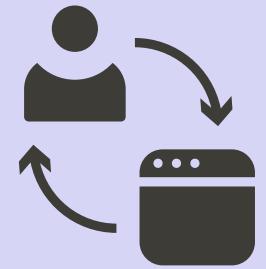
tiny.cc/quizdesign → ETSDESIGN



Vrai/Faux: Il y a 5 opérations système dans ce DSS



DSS



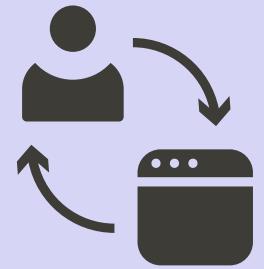
Retour sur l'exercice Google

Voir: Seance-03-solution-reserverLivreBibliotheque

11 . 16

POINTS IMPORTANTS

Voir le PDF des **notes de cours**

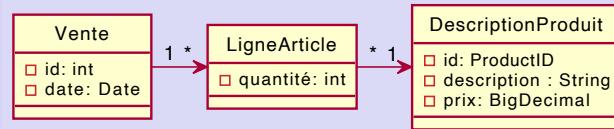


11 . 17

RETOUR D'UNE OPÉRATION SYSTÈME (DSS)



- Du cas d'utilisation...
 - Le système présente le total incluant les taxes calculées.
- L'affichage du total => couche présentation
- Mais le calcul du total?
 - Aucune classe n'en a la responsabilité
 - Affectez-la avec « Expert »
 - (il y a un attribut dérivé / total dans le MDD)



11 . 18

EXPERT POUR CALCULER LE TOTAL



1. Énoncer la responsabilité

- Qui doit connaître le montant total de la vente?

2. Récapituler les informations nécessaires

- total est la somme de tous les sous-totaux des lignes
- sous-total de ligne := quantité * prix

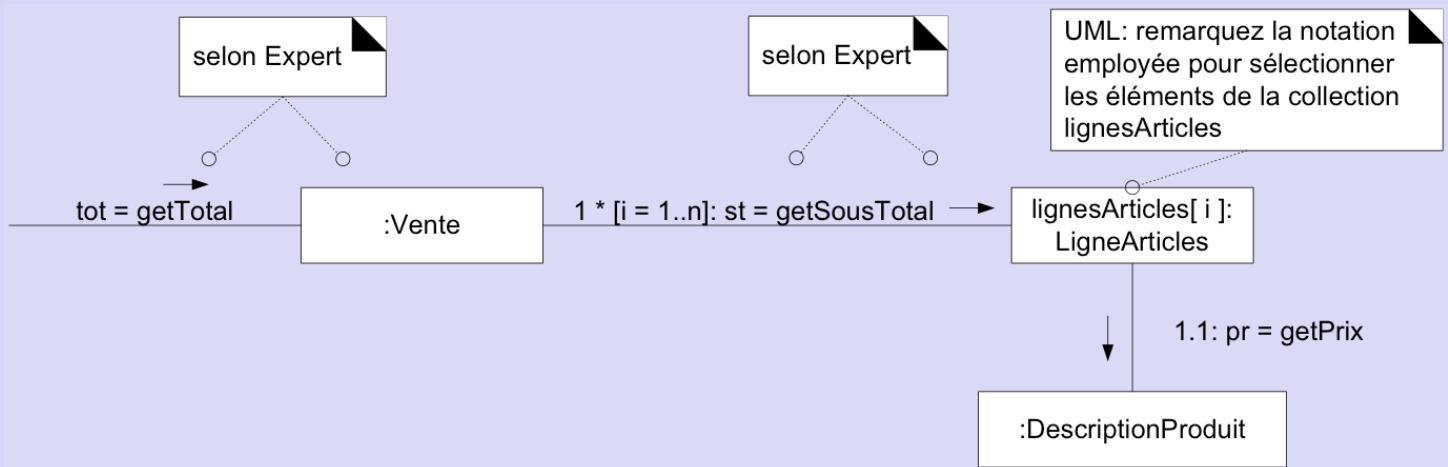
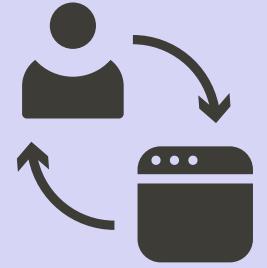
3. Dresser la liste des informations nécessaires et des classes qui les possèdent.

| Informations nécessaires au total de la vente | Expert |
|---|---------------|
| Produit.prix | Produit |
| LigneArticles.quantité | LigneArticles |
| Toutes les LigneArticles de la vente en cours | Vente |

Analyse détaillée: p.F337

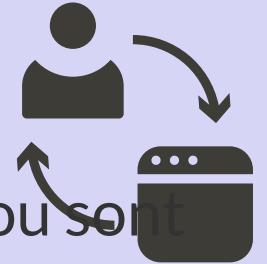
CONCEPTION

Vente.getTotal()



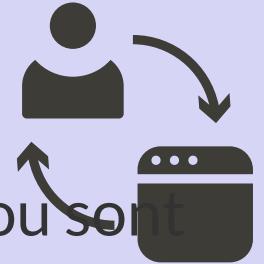
(diagramme de communication)

CHP10 DSS



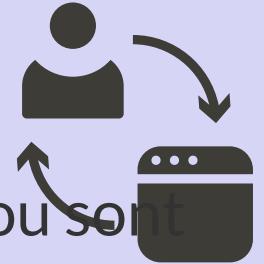
1. Quels sont les artefacts qui influencent ou sont influencés par le DSS?

CHP10 DSS



1. Quels sont les artefacts qui influencent ou sont influencés par le DSS?
2. Qu'est-ce qu'un DSS?

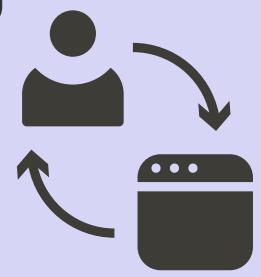
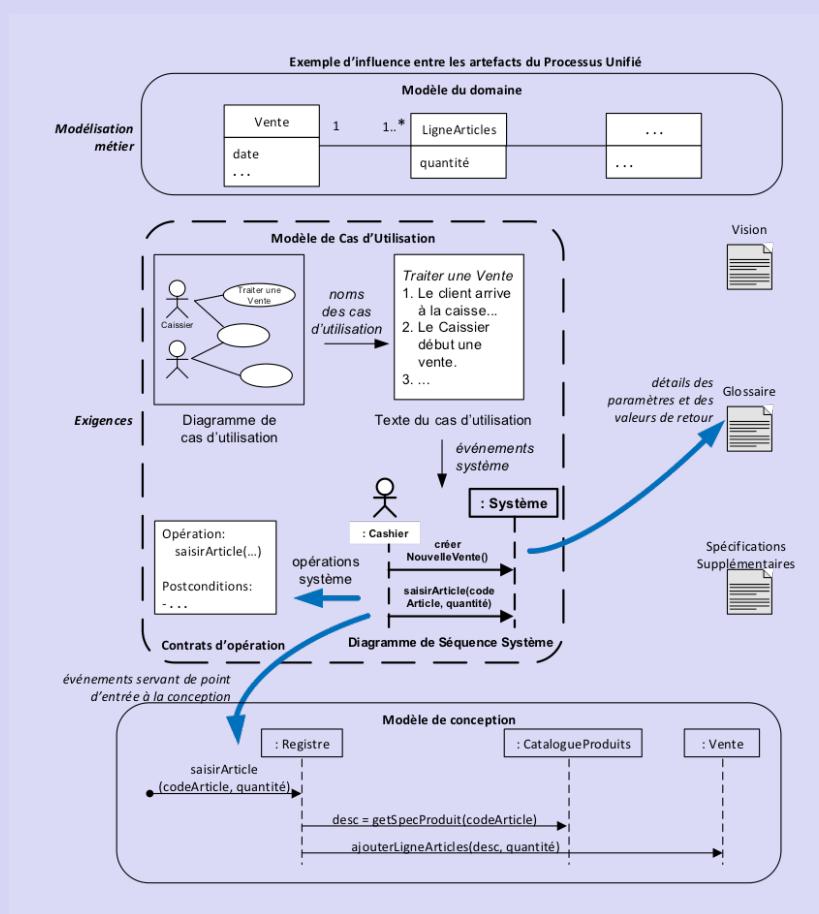
CHP10 DSS



1. Quels sont les artefacts qui influencent ou sont influencés par le DSS?
2. Qu'est-ce qu'un DSS?
3. Pourquoi tracer un DSS?

11 . 21

INFLUENCE ENTRE ARTEFACTS DU PU



MODULE CONTRAT

1. Contrats - Contrats d'opération - 9.57m
2. Contrats d'opération - 12.08m
3. list/map - 4.39m
4. Maps - 1.26[m](https://youtu.be/eSlbIfteQ_M?start=10)
5. Révision - Contrats pour réaliser les RDCU

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

12.1

Contrats



Created by Mohammed Ra

- Transformation au niveau des données dans le MDD suite à l'exécution d'une opération du DSS

12 . 2

Contrats



Created by Mohammed Ra

- Transformation au niveau des données dans le MDD suite à l'exécution d'une opération du DSS
- Expliquer en vos propre mots ce qui se passe au niveau des données de votre modèle du domaine lorsque vous appelez chacune des opérations système.

12 . 2

Contrats



Created by Mohammed Ra

- Transformation au niveau des données dans le MDD suite à l'exécution d'une opération du DSS
- Expliquer en vos propre mots ce qui se passe au niveau des données de votre modèle du domaine lorsque vous appelez chacune des opérations système.
- Opération système == contrat

Contrats d'opération



Created by Mohammed Ra

- Qu'est-ce qu'un contrat d'opération?
- Pourquoi les contrats d'opération?

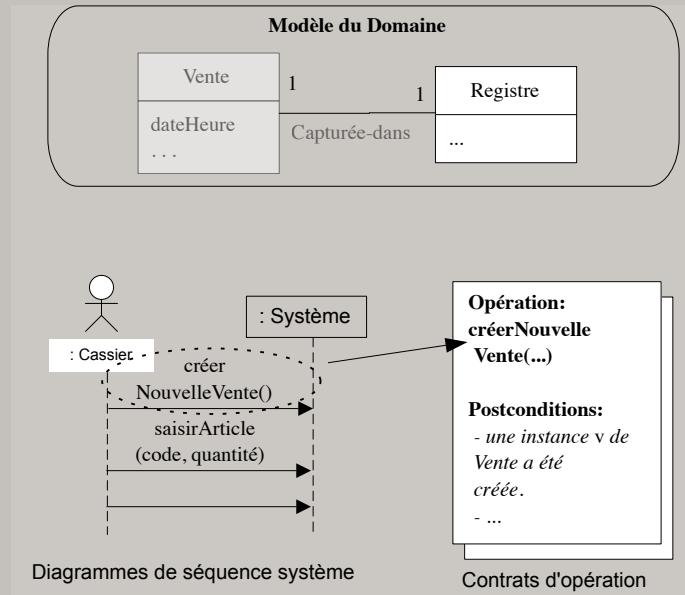
12 . 3

*QU'EST-CE QU'UN CONTRAT D'OPÉRATION?

Un document décrivant ce qui est arrivé après l'exécution d'une opération système.



Created by Mohammed Ra



ÉLÉMENTS D'UN CONTRAT D'OPÉRATION



Created by Mohammed Ra

- Signature de l'opération système:
`saisirArticle(codeArticle: CodeArticle, quantité: int)`
- Postconditions (3 formes, vocabulaire du MDD)
 - création (ou suppression) d'instances;
 - modification des valeurs des attributs;
 - formation (ou rupture) d'associations.

*CONTRAT D'OPÉRATION:



Opération: créerNouvelleVente()

Références croisées: Cas d'utilisation : Traiter Vente

Postconditions:

- une instance v de Vente a été créée
- v a été associée au Registre
- des attributs de v ont été initialisés

Created by Mohammed Ra

12 . 6

POURQUOI LES CONTRATS



Created by Mohammed Ra

- Facilite la conception OO (RDCU)
 - Liste d'épicerie (quoi faire)
 - RDCU s'inspire du MDD (réduire décalage des représentations)
- Donne des conditions pour tests
- Aide à valider le MDD

12 . 7

EXERCICE CONTRATS (GOOGLE CLASSROOM)



Created by Mohammed Ra

Voir seance-03-exercice-reserverLivre

12 . 8

CONTRATS



Quel sont les trois type de postcondition

Created by Mohammed Ra

12 . 9

CONTRATS



Created by Mohammed Ra

- Postcondition:
 - « ... sur la base de correspondance avec clé. »
 - implique une multiplicité de 1 à plusieurs
 - implique l'utilisation d'une Map (tableau associatif)

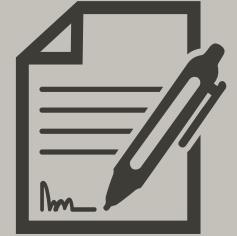
CONTRATS - MAPS



Created by Mohammed Ra

- sur la base de correspondance avec “Clé”
 - implique l’utilisation d’un tableau associatif (Map<>)

CHP11 CONTRATS



1. Quels sont les éléments d'un contrat?

Created by Mohammed Ra

12 . 12

CHP11 CONTRATS



1. Quels sont les éléments d'un contrat?
2. Quelle est la relation entre le DSS et les contrats?

Created by Mohammed Ra

12 . 12

CHP11 CONTRATS



Created by Mohammed Ra

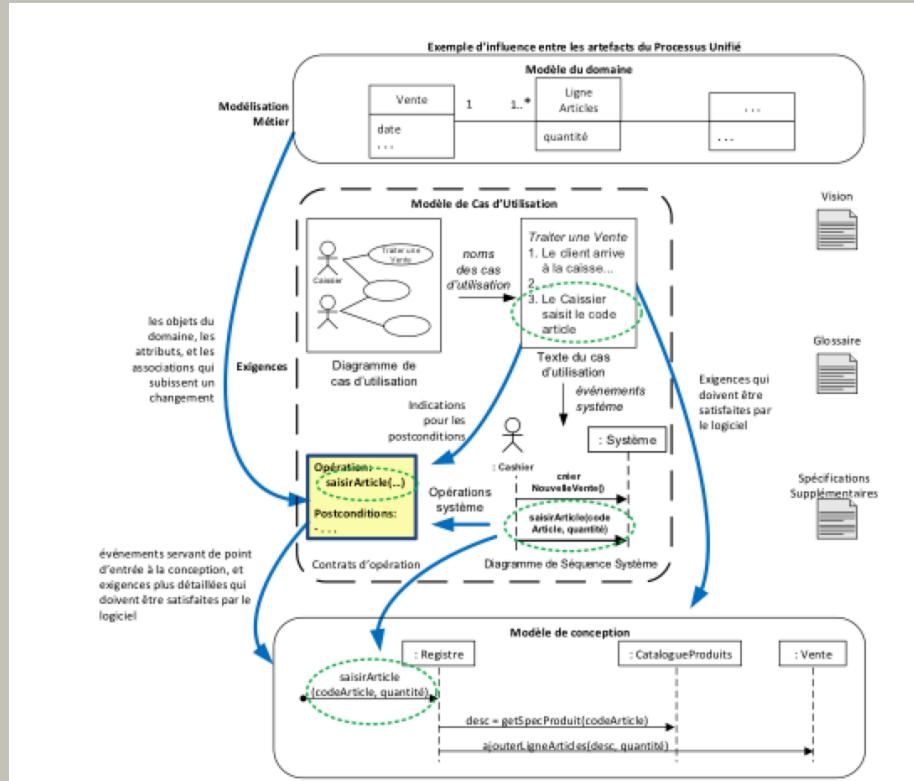
1. Quels sont les éléments d'un contrat?
2. Quelle est la relation entre le DSS et les contrats?
3. Que décrivent les postconditions?

12 . 12

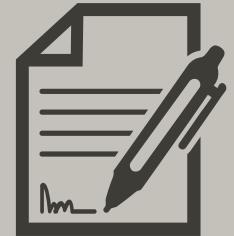
INFLUENCE D'ARTEFACTS DU PROCESSUS UNIFIÉ



Created by Mohammed Ra



CONTRATS - ERREURS FRÉQUENTS



Created by Mohammed Ra

- Ne pas mettre les paramètres de l'opération
- Inventer de nouveaux paramètres
- Ne pas utiliser un paramètre pour trouver une instance d'un objet lorsque le contrat spécifie que la relation est faite sur la base de correspondance avec une clé.
- Les postconditions ne sont pas écrites au passé.
- Les postconditions ne sont pas cohérentes avec le MDD

MODULE RDCU

1. Réalisation d'un cas d'utilisation - 25.10m
2. GRASP - Contrôleur - 17.54m
3. GRASP - Créateur et Expert - 24.28m
4. GRASP - Couplage et Cohésion - 27.49m
5. Exercice avancé - 30.33m
6. Révision - 9.42m
7. GRASP - polymorphisme, fabrication pure, indirection, protection des variations - 19.40m
8. GRASP sont une généralisation d'autres patterns - 16.43m
9. Patron faire soi-même

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

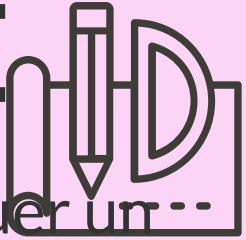
RÉALISATION D'UN CAS D'UTILISATION



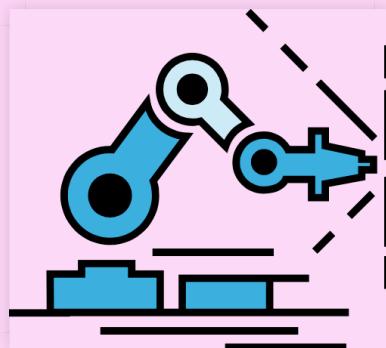
- Un diagramme d'interaction avec des annotations pour réaliser une opération système et satisfaire:
 - Son contrat
 - Ses retours d'informations
- Deux types de responsabilités
 - Faire
 - Savoir

<https://plantuml.com/sequence-diagram>

RESPONSABILITÉS DE FAIRE

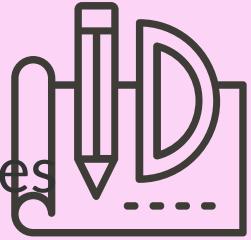


- faire quelque chose lui-même (ex. effectuer un calcul)
- déclencher une action d'un autre objet
- contrôler et coordonner les activités d'autres objets



13 . 3

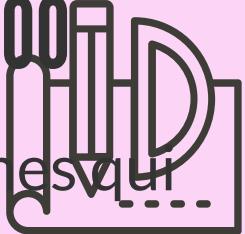
RESPONSABILITÉS DE SAVOIR



- connaître les données privées encapsulées
- connaître les objets connexes
- connaître des éléments qu'il peut dériver ou calculer



LA CPR EST UNE MÉTAPHORE POUR LA COOPÉRATION

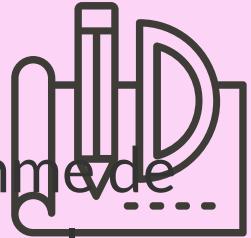


- On y assimile les objets à des personnes qui
 - ont des responsabilités
 - collaborent avec d'autres personnes
 - réalisent une tâche



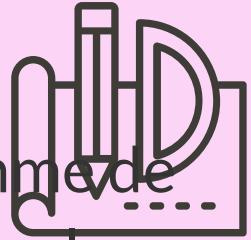
CPR = Conception pilotée par les responsabilités

RÉALISER VOS PREMIERS RDCU



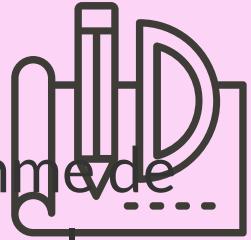
- Concrètement: Un RDCU est un diagramme de séquence ou de communication qui réalise le contrat associer à l'opération système et s'assure de pouvoir retourner les informations demandées.

RÉALISER VOS PREMIERS RDCU



- Concrètement: Un RDCU est un diagramme de séquence ou de communication qui réalise le contrat associer à l'opération système et s'assure de pouvoir retourner les informations demandées.
- Créer un contrôleur

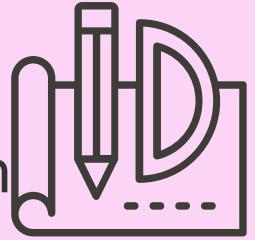
RÉALISER VOS PREMIERS RDCU



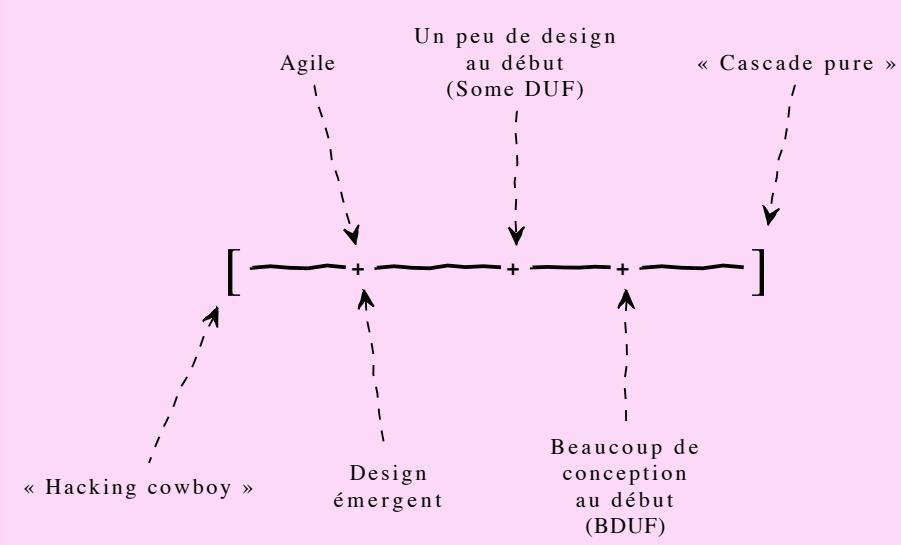
- Concrètement: Un RDCU est un diagramme de séquence ou de communication qui réalise le contrat associer à l'opération système et s'assure de pouvoir retourner les informations demandées.
- Créer un contrôleur
- Faire un diagramme de séquence ou de communication pour chaque opération système.



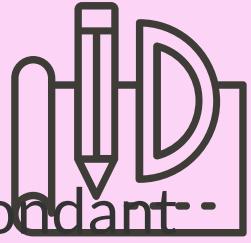
:POURQUOI FAIRE UNE RDCU?



Pour apprendre à faire une solution
modulaire et intuitive

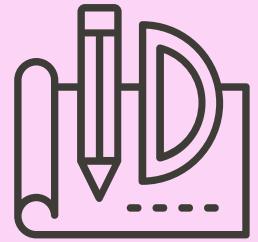


RDCU: ASPECTS



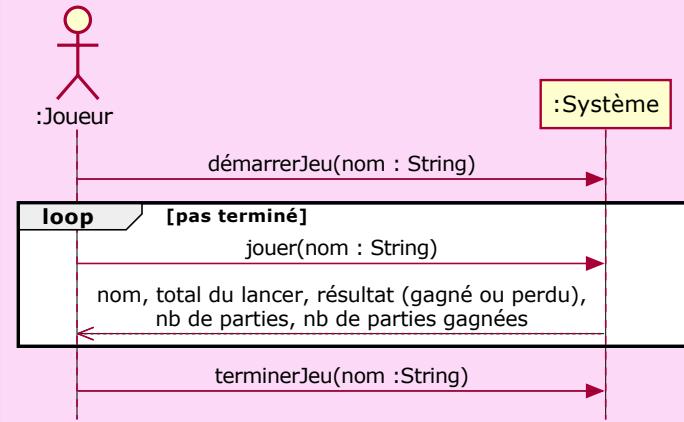
- Proposer des **classes logicielles** correspondant aux **classes conceptuelles**
 - Inspirées du MDD
 - Pour réduire le *décalage des représentations*
- Utiliser les principes GRASP
 - **Faible Couplage et Forte cohésion** (LOG121!)
 - **Contrôleur** (architecture en couches)

RDCU - JEU DE DÉS

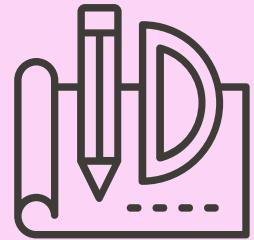


Opérations système du DSS

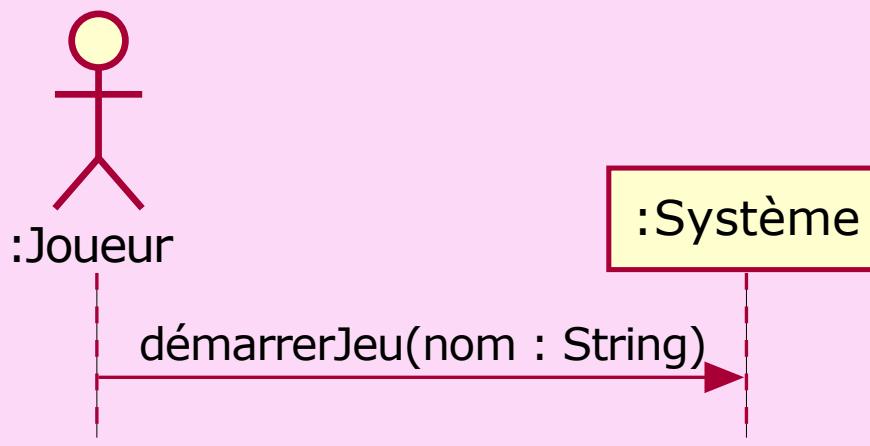
DSS pour un scénario adapté de *Jouer aux dés*
(Ch. 1 de Larman)



RDCU - JEU DE DÉS



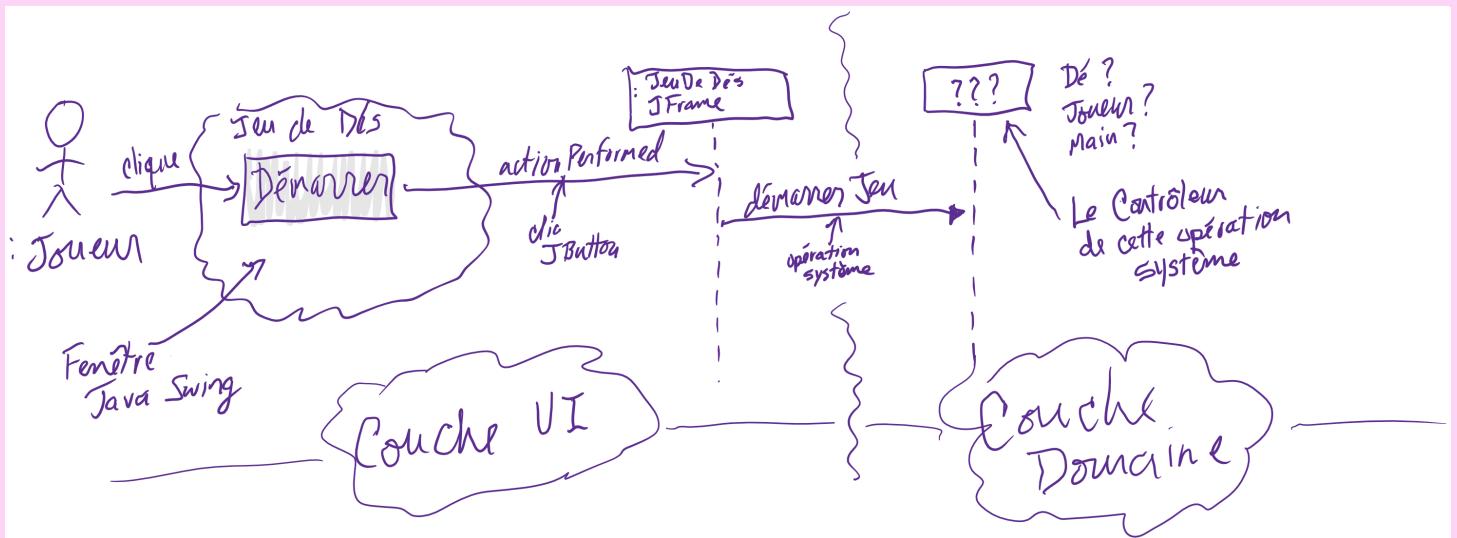
Première opération système:



Qui envoie l'opération? Qui la reçoit?

RDCU (SOLUTION JAVA SWING)

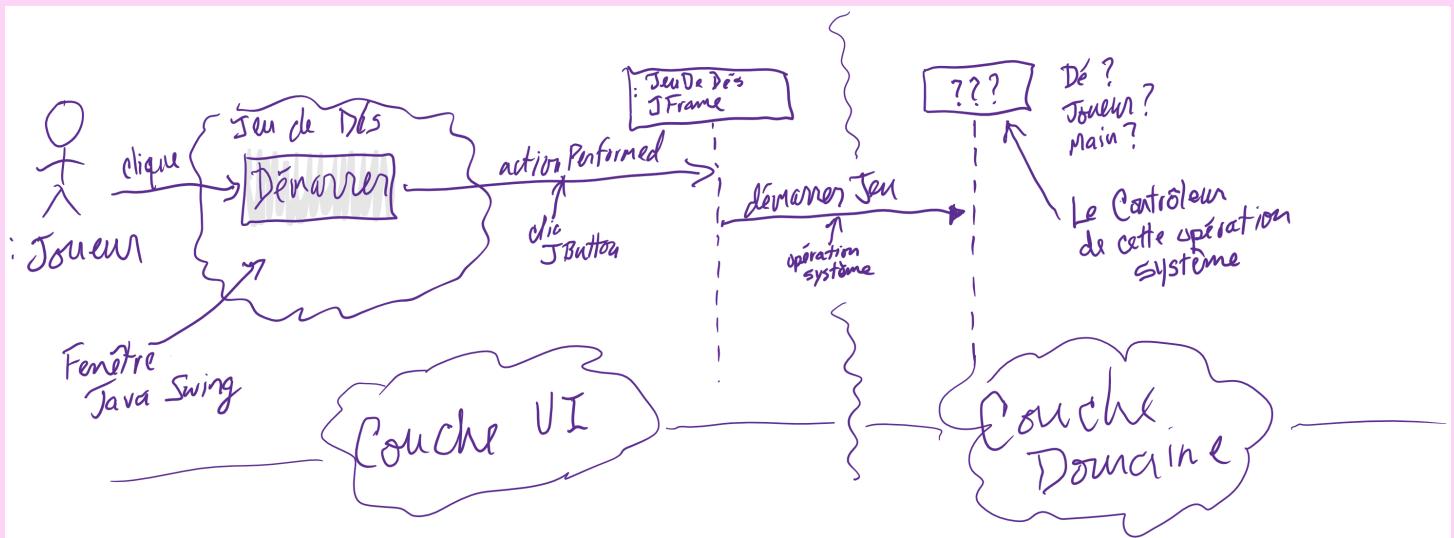
Opération démarrerJeu - qui envoie cette opération?



RDCU (SOLUTION JAVA SWING)



Opération démarrerJeu - qui reçoit cette opération?



PRINCIPE CONTRÔLEUR GRASP



Problème: Quel est le premier objet en dehors de la couche présentation qui reçoit et coordonne (« contrôle ») les opérations système ?

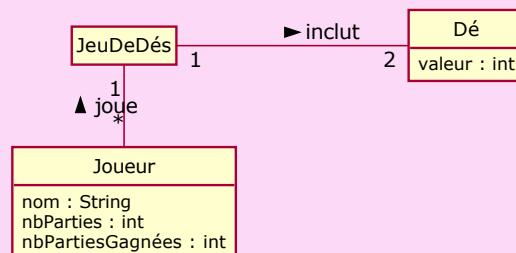
PRINCIPE CONTRÔLEUR GRASP



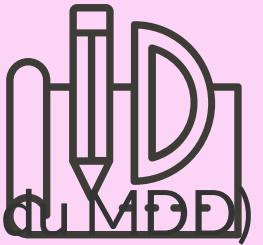
Solution: Affectez une responsabilité à la classe qui correspond à l'une de ces définitions:

1. Elle représente le **système global**, un « **objet racine** », un **équipement** ou un **sous-système**.
2. ...

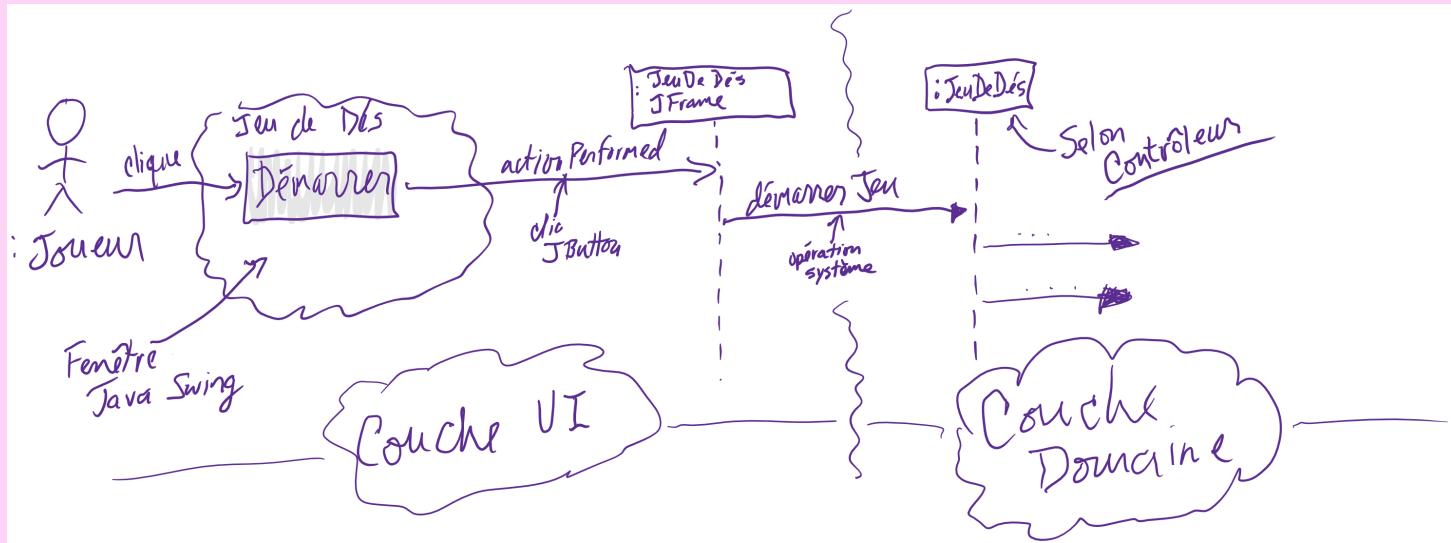
Modèle du domaine (adapté du Jeu de dés du Ch. 1 de Larman)



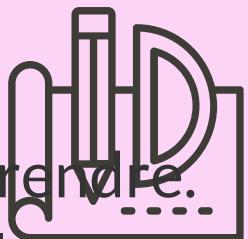
RDCU - CONTRÔLEUR



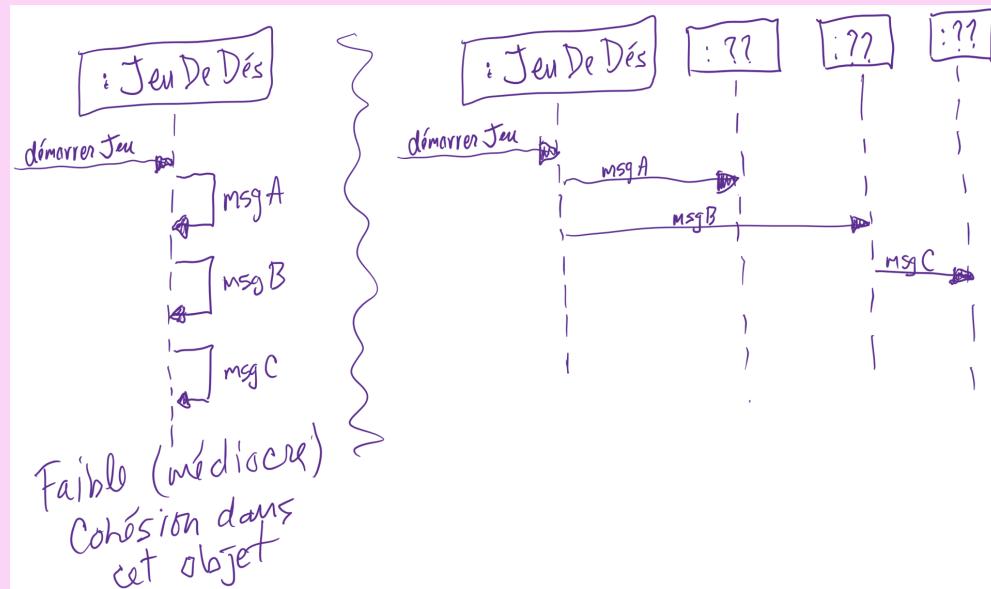
JeuDeDés est le contrôleur GRASP (inspiré du MDD)



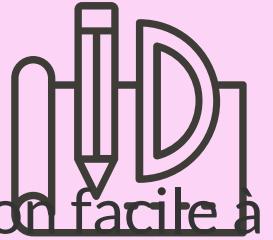
RDCU - DÉTAILLÉ



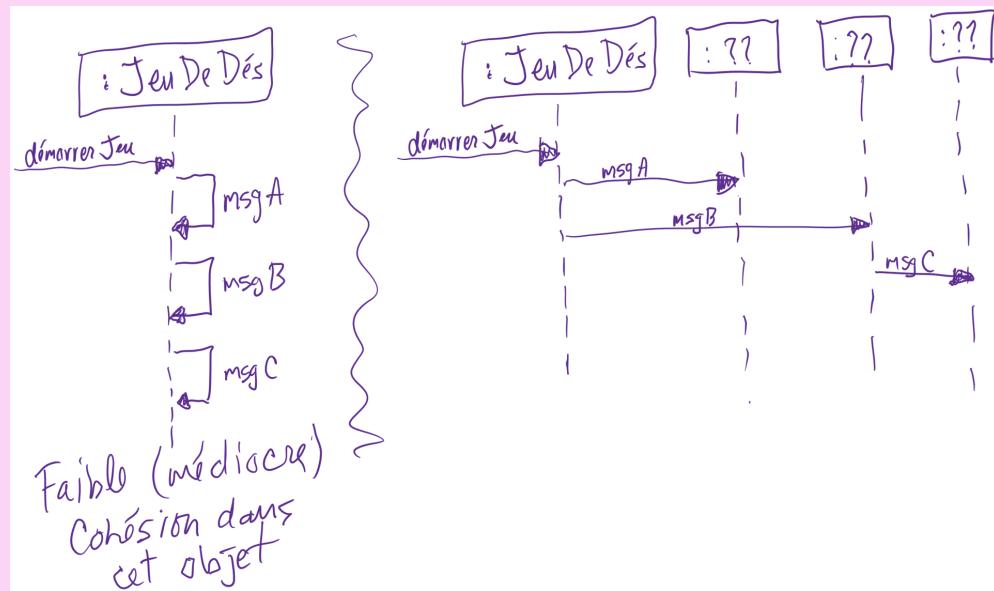
Faire un design modulaire et facile à comprendre.
Affecter les responsabilités aux bonnes classes.



RDCU



Prendre les bonnes décisions pour une solution facile à comprendre et modulaire...



DÉCALAGE DES PRÉSENTATIONS



Facile? Les classes logicielles devraient ressembler à des classes conceptuelles.

Qui fait quoi? Qui a quelle responsabilité?

RDCU



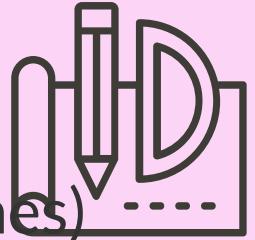
Approche: conception orientée-responsabilités--

GRASP

General Responsibility Assignment Software Patterns

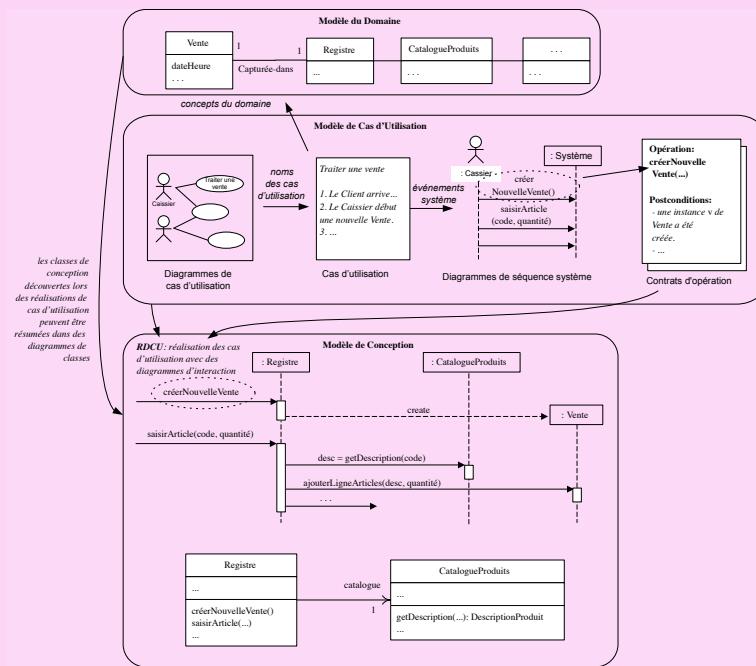
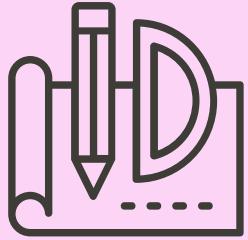
Pour décider où mettre les méthodes...

GRASP

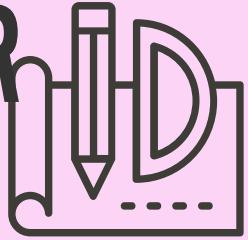


- Contrôleur (séparation des couches) -----
- **Créateur**
- **Expert en information**
- Faible couplage
- Forte cohésion
- Polymorphisme
- Indirection
- Protection de variation
- Fabrication pure

*RDCU (SURVOL)



RDCU: SCÉNARIO DÉMARRER

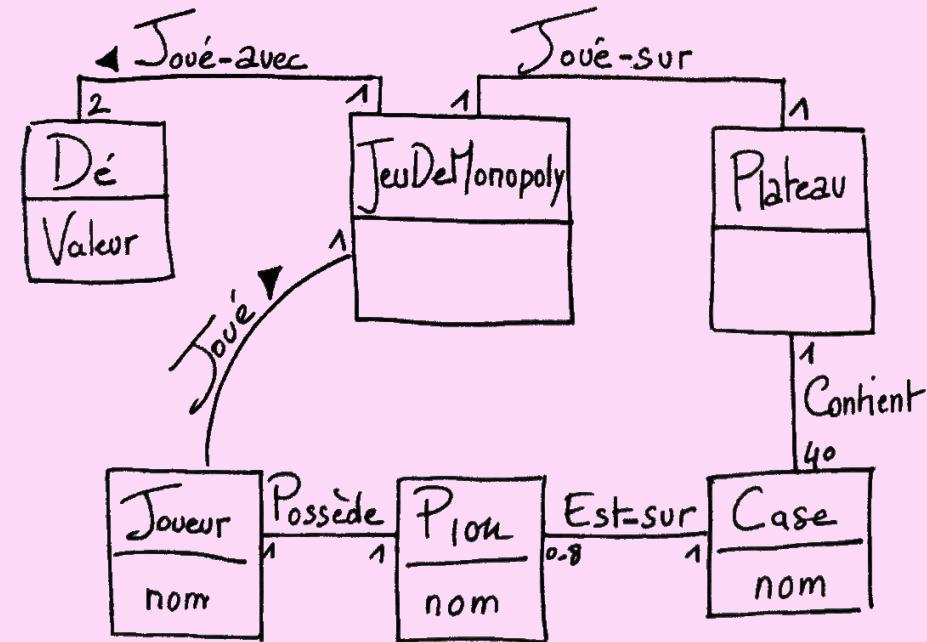


- C'est l'initialisation du système.
- C'est implicite mais essentiel!
- On doit instancier les objets faisant partie de l'application.

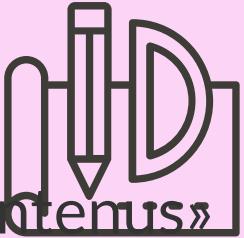
INSTANCIER LES OBJETS CASE



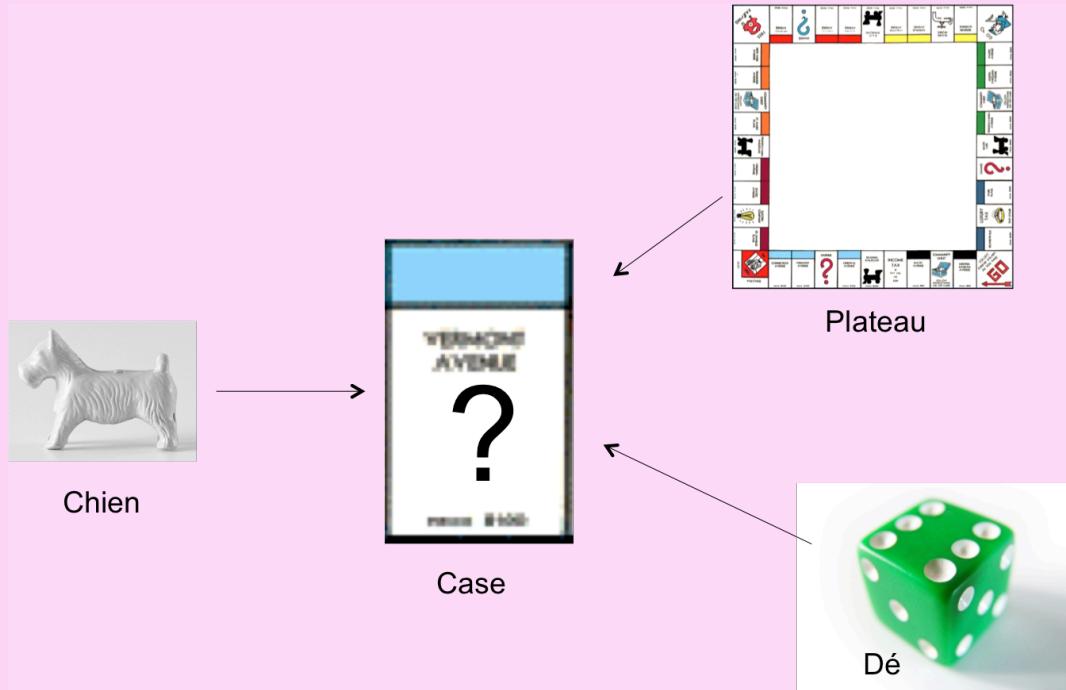
- Quelle classe s'en occupe? (sondage ~~Zoom~~)--



CRÉATEUR (GRASP)



- Les «conteneurs» créent les objets «contenus»



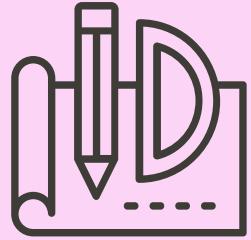
CRÉATEUR (GRASP)



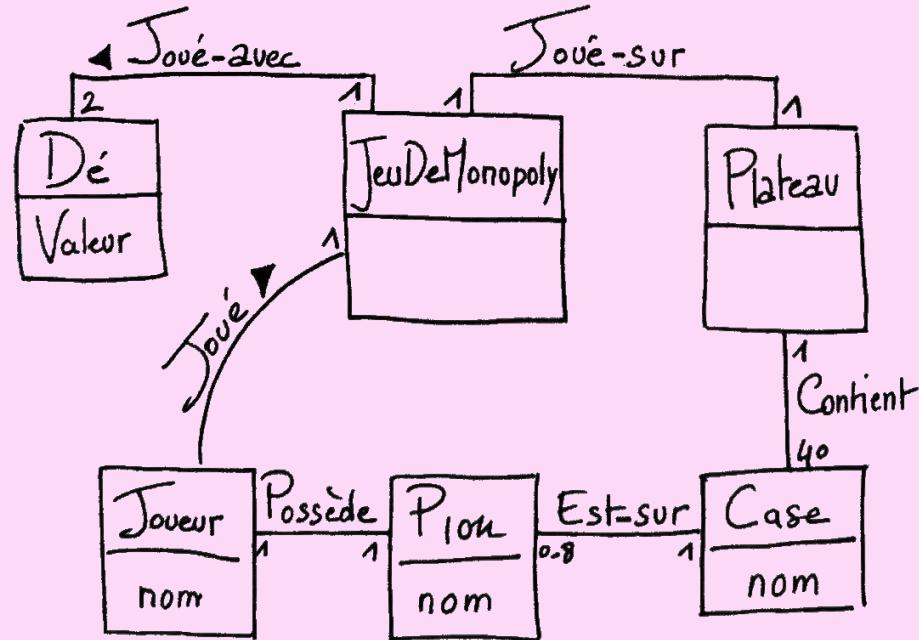
- **Problème:** Qui crée? (postcondition d'un contrat)
- **Solution:** Affecter à la classe B la responsabilité de créer les objets d'une classe A si...
 - B possède les données d'initialisation des objets A
 - B contient ou agrège des objets A
 - B utilise étroitement des objets A
 - B enregistre des objets A

On s'inspire du MDD. On réutilise les liens existents.

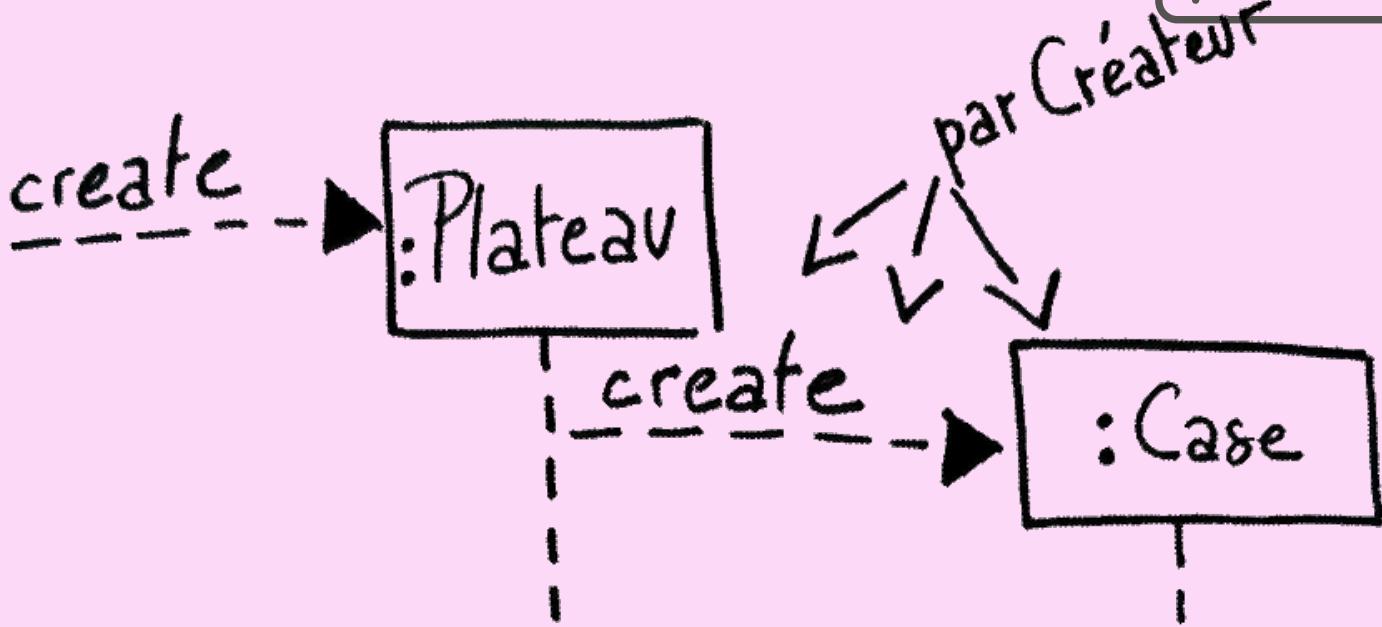
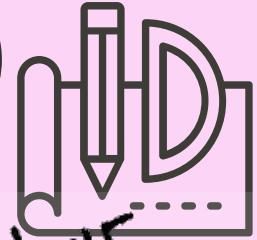
*CRÉATEUR



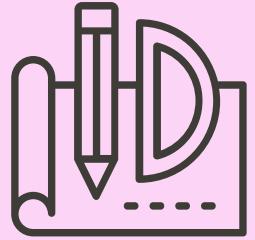
- Qui crée les cases (Square)?



*CRÉATEUR (ANNOTATION)



EXPERT EN INFORMATION



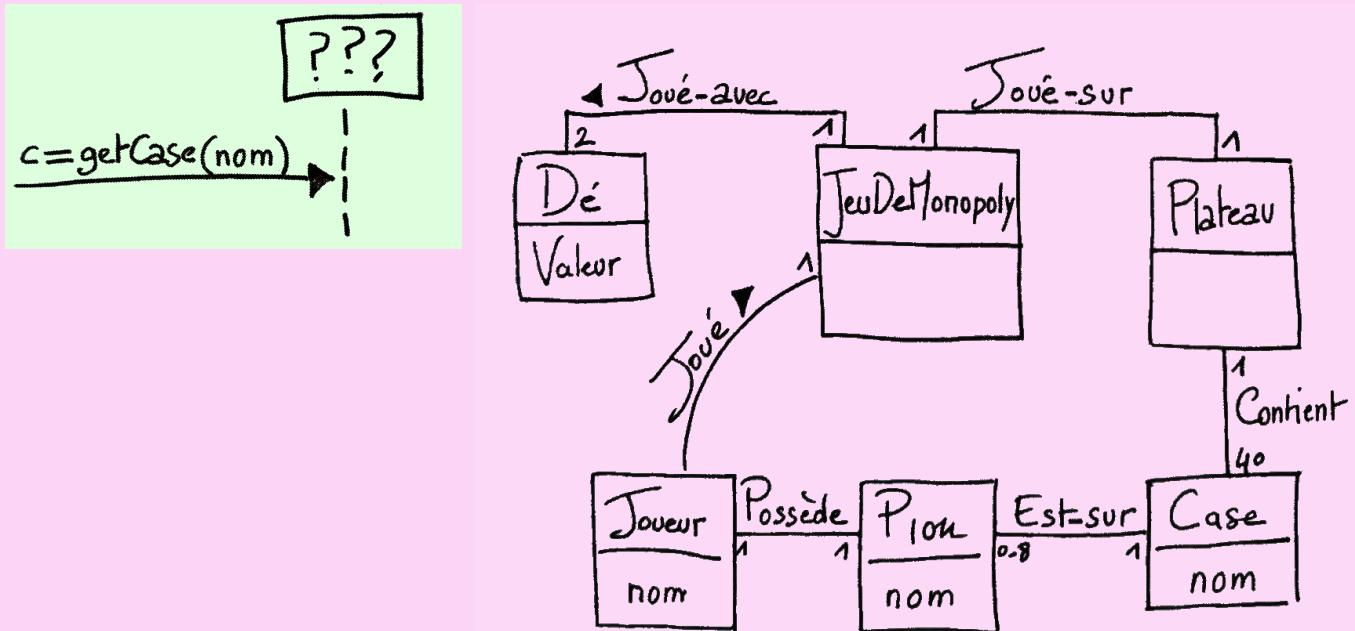
- **Problème:** Quel est le principe général d'affectation des responsabilités aux objets?
- **Solution:** Affecter la responsabilité à la classe qui possède les informations nécessaires pour s'en acquitter

En termes de paramètres, associations

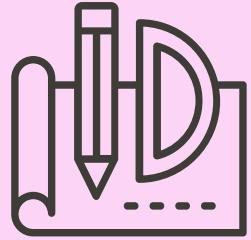
EXPERT (GRASP)



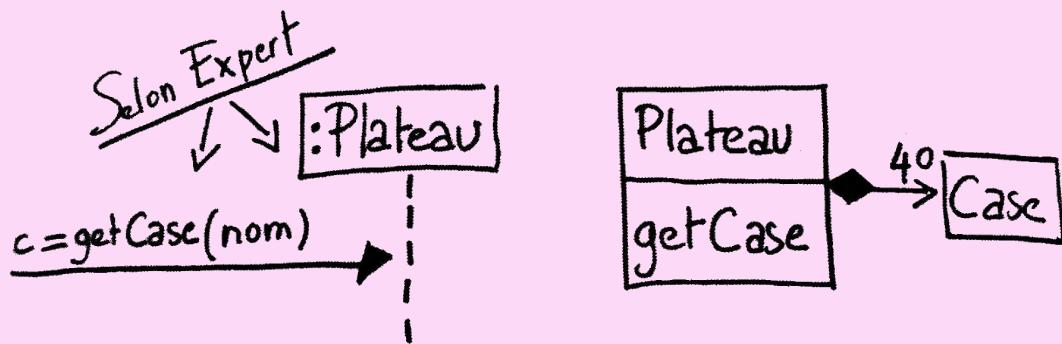
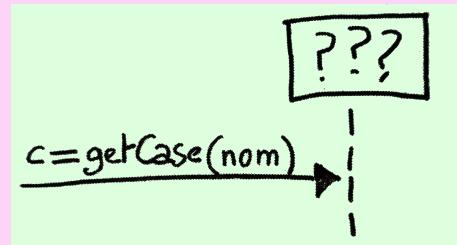
- (Sondage) Où mettre la méthode `getCase(nom)` ?



EXPERT

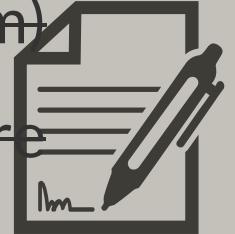


- Application du patron Expert



~~## Exercice RDCU (Google Classroom)~~

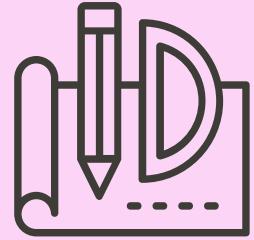
~~Voir seance 03 exercice reserver~~



Created by Mohammed Ra

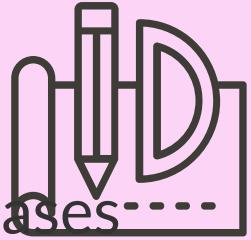
13 . 31

GRASP

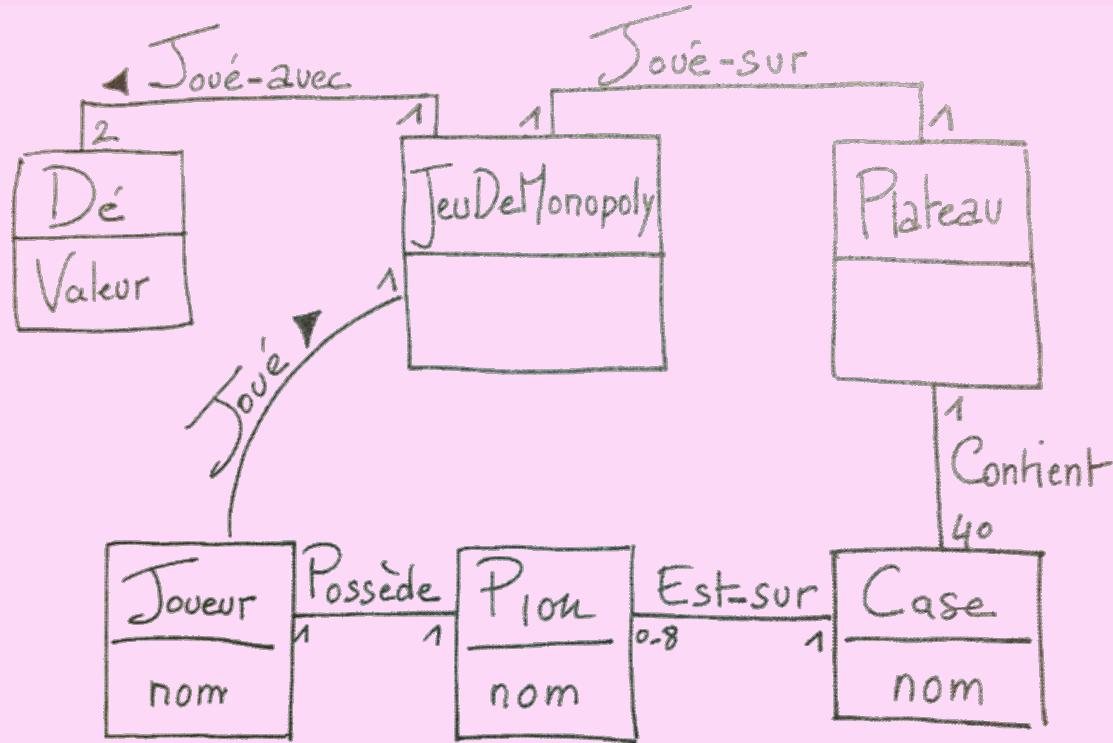


- Contrôleur
- Créateur
- Expert en information
- **Faible couplage**
- **Forte cohésion**
- Polymorphisme
- Indirection
- Protection de variation
- Fabrication pure

FAIBLE COUPLAGE



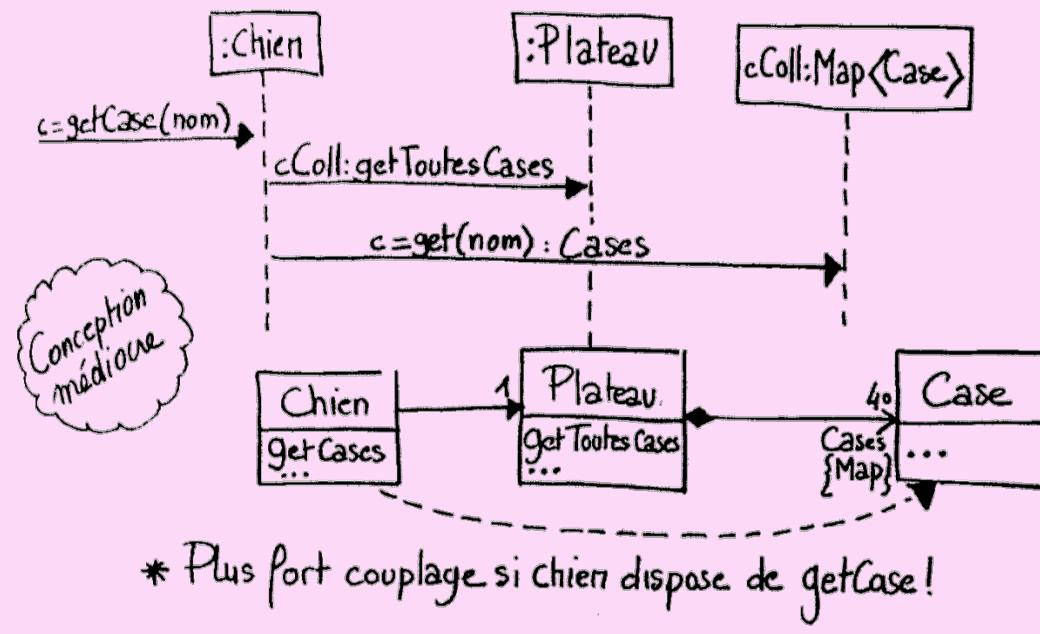
Question : Qui peut fournir la liste des cases----



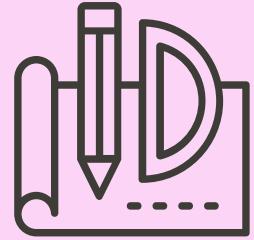
FAIBLE COUPLAGE



- Si la classe Pion (chien) avait la responsabilité:



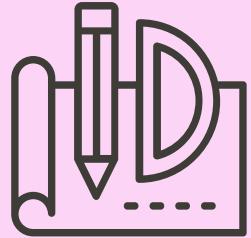
FAIBLE COUPLAGE



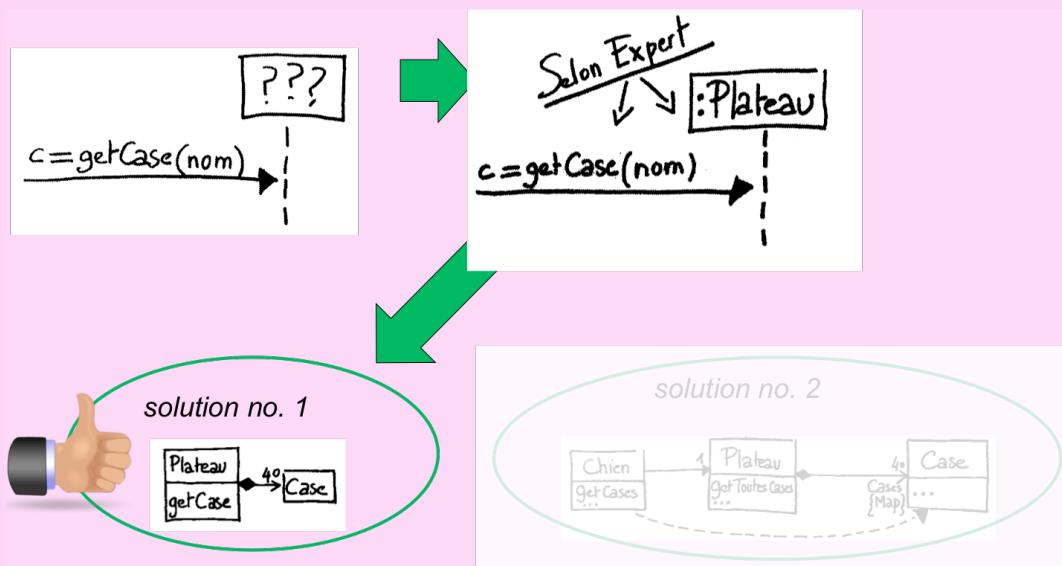
- Problème :
 - Comment réduire l'impact des modifications?
- Solution :
 - Assigner les responsabilités de sorte à éviter tout couplage inutile.
 - Appliquer ce principe pour évaluer plusieurs solutions possibles.



FAIBLE COUPLAGE

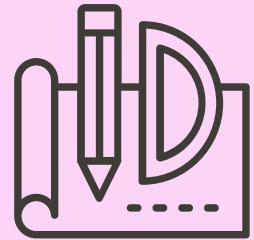


- Expert soutient Faible Couplage
- Expert évite de faire du couplage inutile, puisqu'un expert n'aura pas à demander à quelqu'un d'autre...



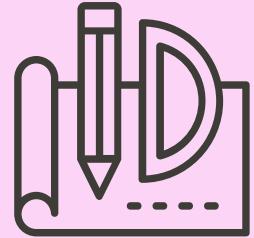
FAIBLE COUPLAGE

MOTIVATION

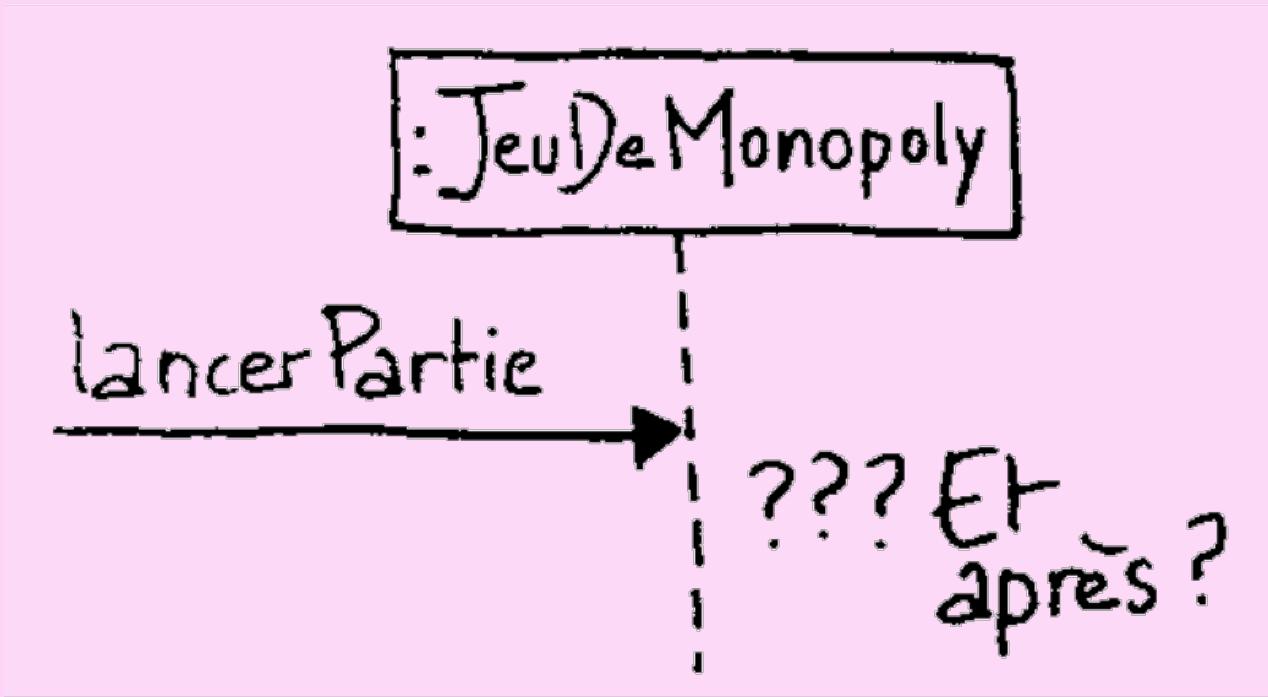


- Classes fortement couplées sont
 - difficiles à maintenir
 - difficiles à comprendre seules
 - difficiles à réutiliser

FORTE COHÉSION

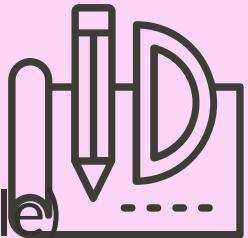


- Poursuivons la conception



13 . 38

COHÉSION

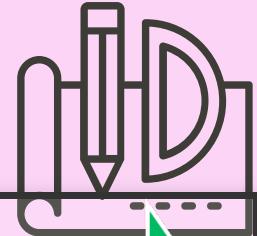


- Mesure de combien une classe (ou module) supporte une seule responsabilité



NF Photography 2013

COHÉSION



Faible cohésion

Forte cohésion

Faible

Forte

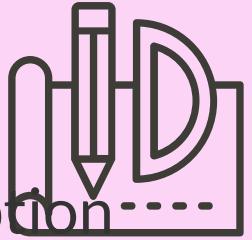
Beaucoup de fonctions

Peu de fonctions

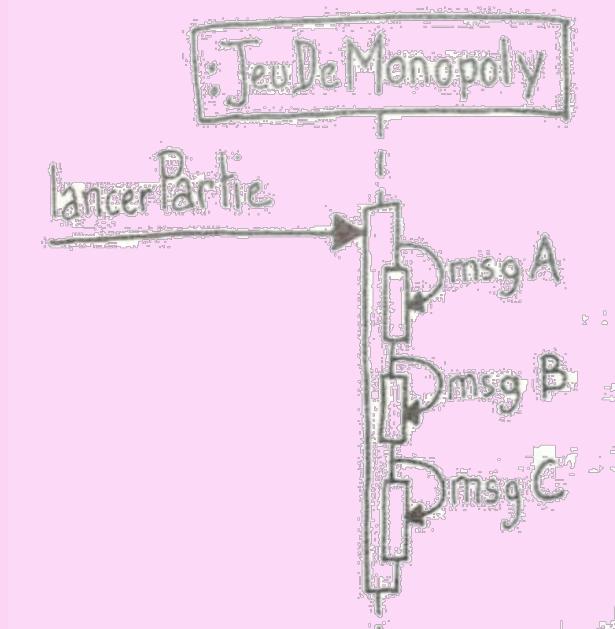
Responsabilités écartées

Fonctions apparentées

FORTE COHÉSION



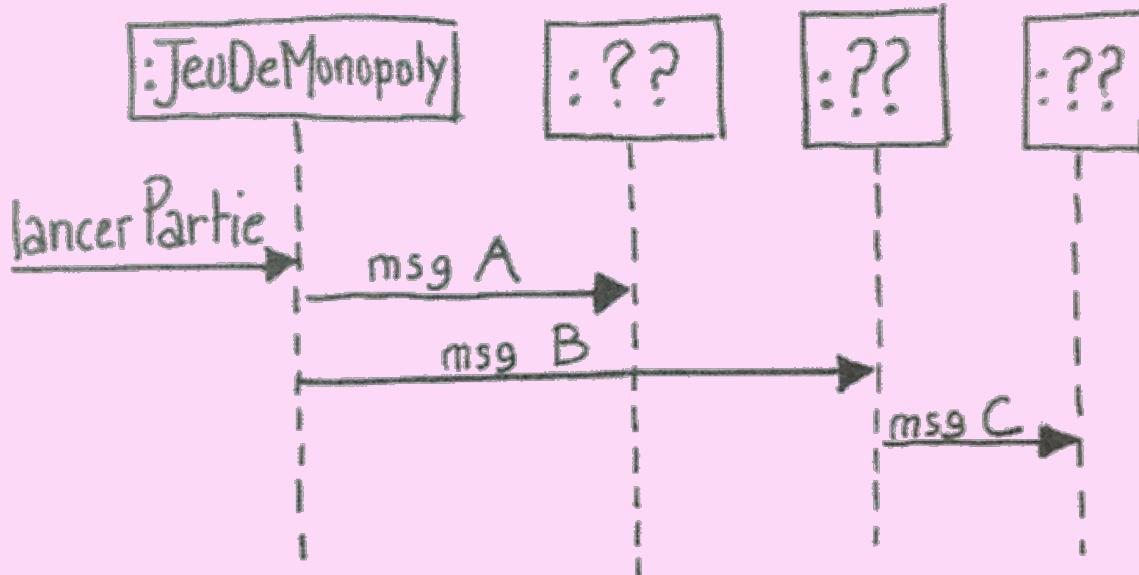
- Plusieurs choix opposés de conception



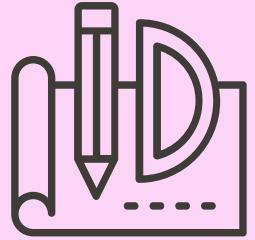
FORTE COHÉSION



- JeuDeMonopoly délègue les responsabilités -



FORTE COHÉSION

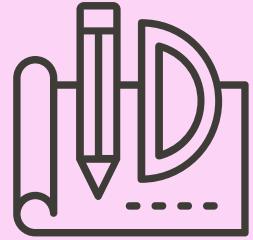


1. Problème :

- Comment s'assurer que les objets restent compréhensibles et faciles à gérer, et, bénéfice second, qu'ils contribuent au Faible Couplage?

2. Solution :

FORTE COHÉSION



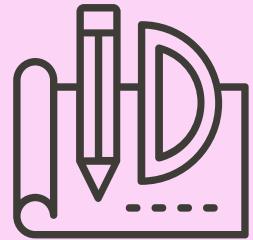
1. Problème :

- Comment s'assurer que les objets restent compréhensibles et faciles à gérer, et, bénéfice second, qu'ils contribuent au Faible Couplage?

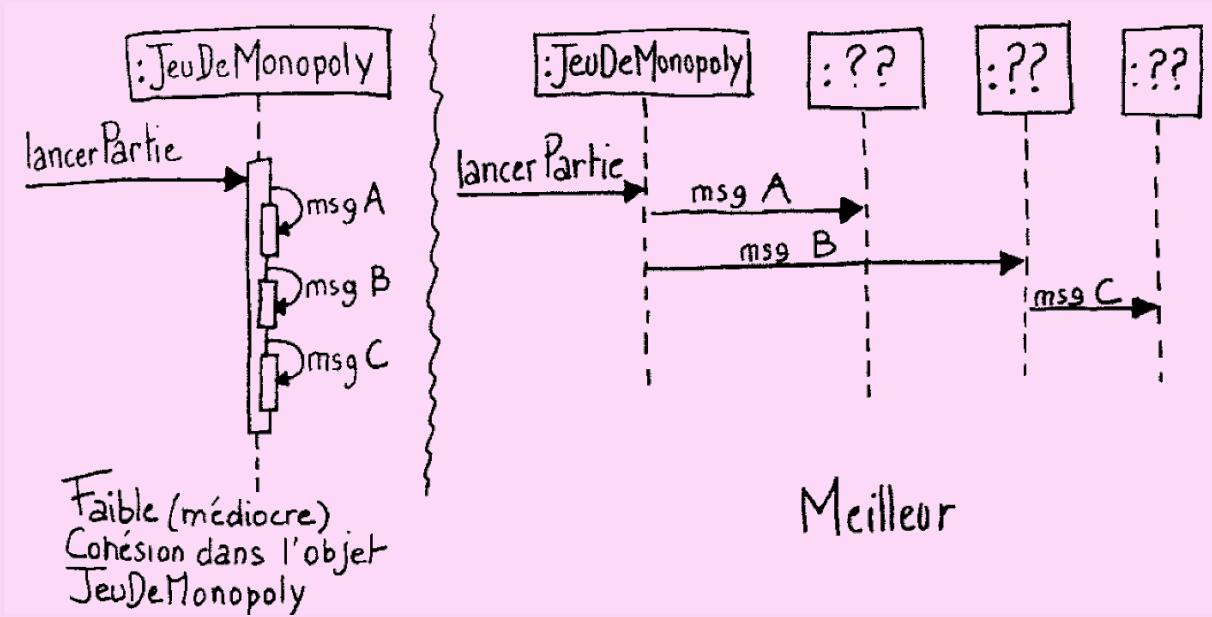
2. Solution :

- Assigner les responsabilités de sorte que la cohésion demeure élevée. Appliquer ce principe pour évaluer les solutions possibles.

FORTE COHÉSION



- Comparaison des choix



COHÉSION: EXERCICE



- Lesquelles des classes suivantes ont de multiples responsabilités?

Jeu

- login()
- inscrire()
- déplacer()
- tirer()
- rester()

Joueur

- setNom()
- setAdresse()
- setNuméroTéléphone()
- sauvegarder()
- charger()

Téléphone

- composer()
- raccrocher()
- parler()
- envoyerDonnées()
- flash()

JeuCartes

- hasNext()
- next()
- remove()
- addCard()
- removeCard()
- shuffle()

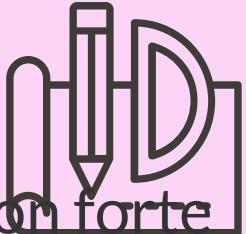
Panier

- ajouter()
- enlever()
- encaisser()
- sauvegarder()

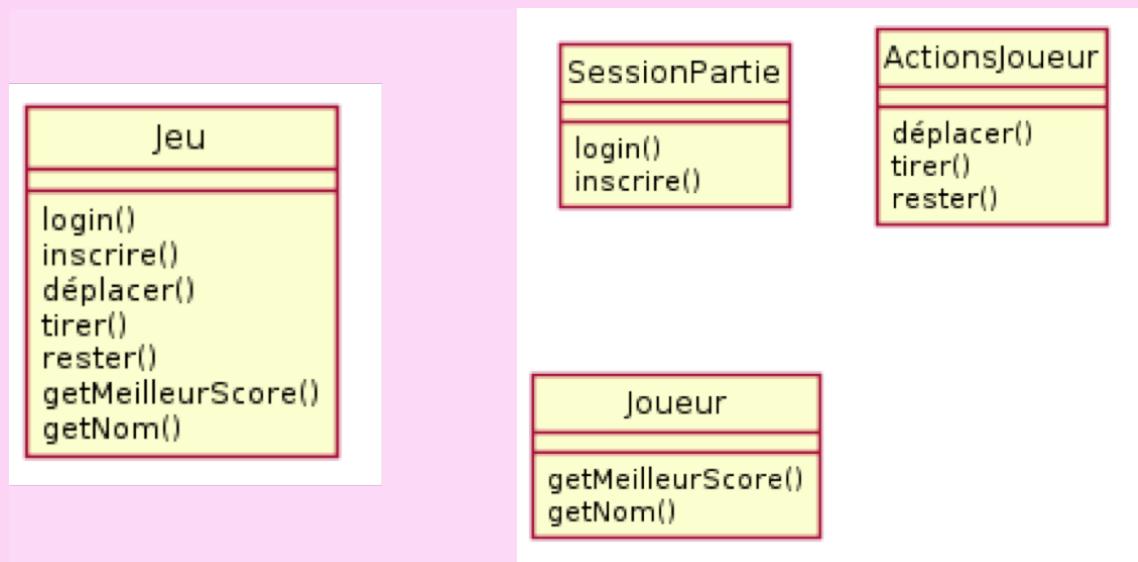
Iterator

- hasNext()
- next()
- remove()

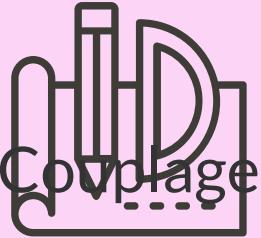
COHÉSION: EXERCICE



- Déterminez si ces classes ont une cohésion forte ou faible:



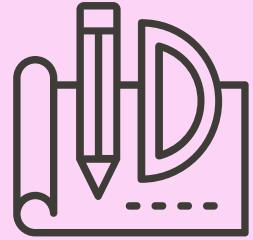
VALIDATION DE LA COMPRÉHENSION



Les patrons GRASP Forte Cohésion et Faible Couplage servent principalement à

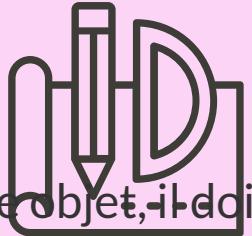
1. concevoir des classes avec beaucoup de responsabilités
2. choisir les classes dans la couche applicative pour traiter une opération système
3. évaluer plusieurs possibilités lors de la réalisation d'une dynamique de conception
4. choisir une bonne classe pour créer des instances d'une autre classe

RDCU-GRASP

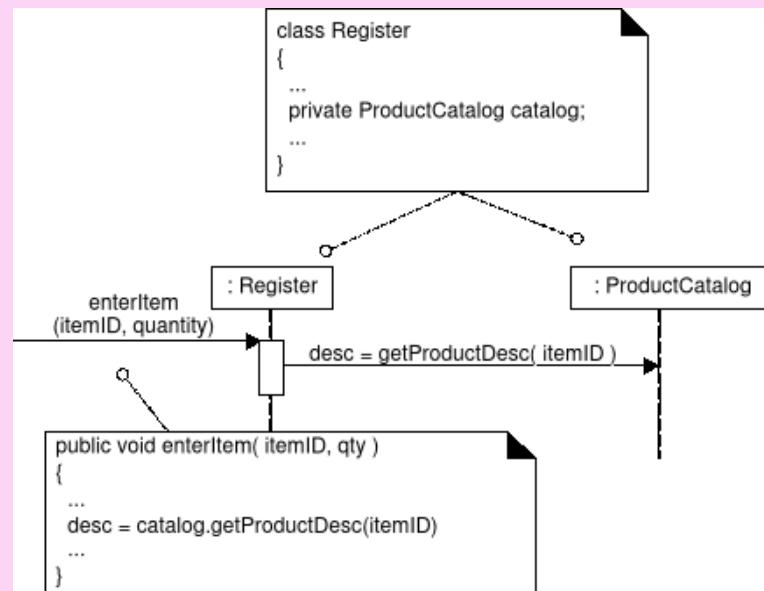


1. Contrôleur
2. Créateur
3. Expert en information
4. Faible couplage
5. Forte cohésion
 - Polymorphisme
 - Indirection
 - Protection de variation
 - Fabrication pure

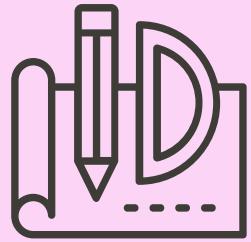
VISIBILITÉ



- Avant qu'un objet puisse envoyer un message à un autre ~~objet, il doit « voir » ce dernier~~
- Avoir une visibilité = posséder une référence (en Java) F18.1

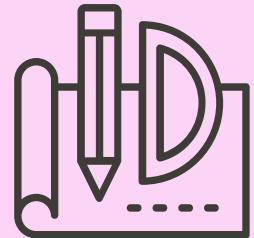


TYPES DE VISIBILITÉ

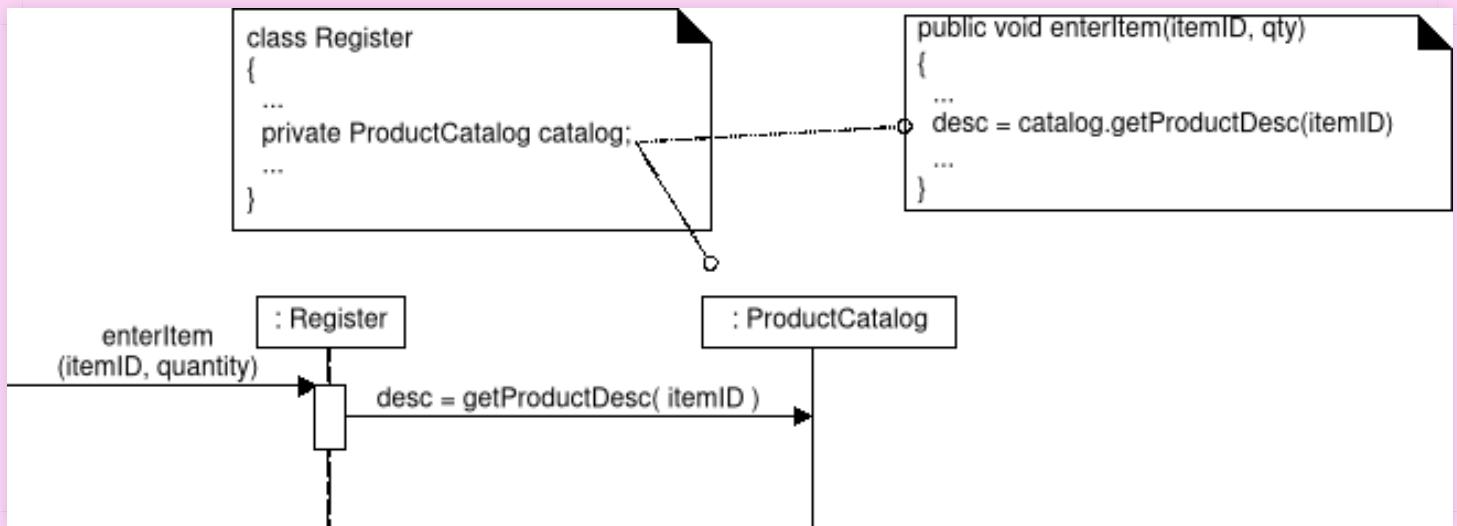


- Attribut
- Paramètre
- Locale
- Globale

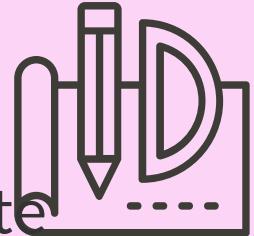
VISIBILITÉ D'ATTRIBUT



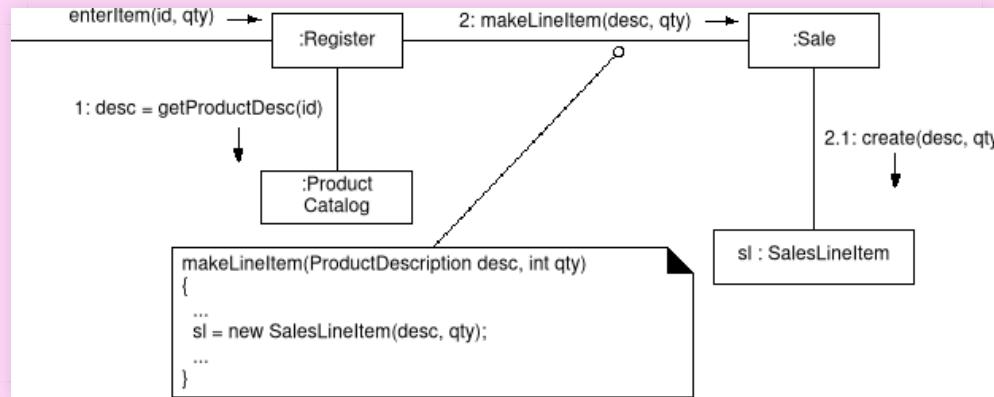
- Relativement permanente
- Très commune



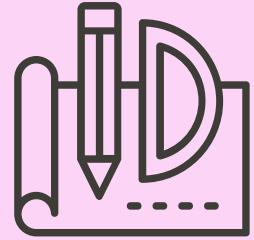
VISIBILITÉ DE PARAMÈTRE



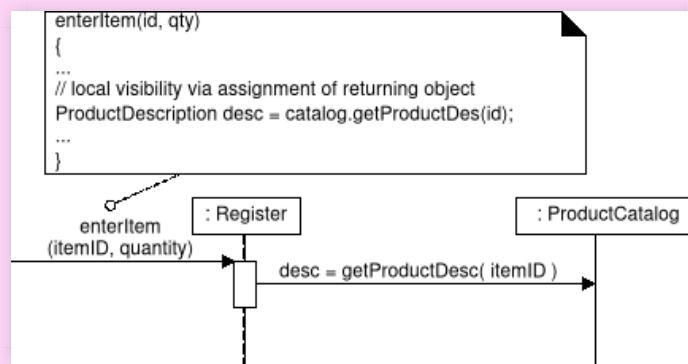
- Durée de vie relativement courte
- Paramètre peut devenir attribut



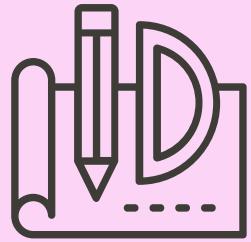
VISIBILITÉ LOCALE



- Durée de vie relativement courte
- Peut être définie via
 - Instance créée localement
 - Instance retornnée par une méthode



VISIBILITÉ GLOBALE



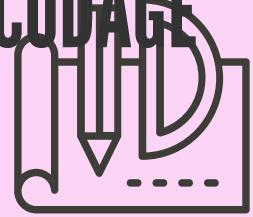
- Relativement permanente
- Moins commune
- Meilleure définition via un singleton (GoF)

```
public class LogManager {  
    private java.io.PrintStream sout ;  
    private static LogManager lmInstance ;  
    private LogManager ( java.io.PrintStream out )  
    {  
        sout = out;  
    }  
    public void log( String msg )  
    {  
        sout.println ( msg );  
    }  
    public static LogManager getInstance () {  
        if ( lmInstance == null ) {  
            lmInstance = new LogManager ( System.out );  
        }  
        return lmInstance ;  
    }  
}
```

Usage:

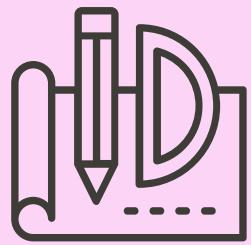
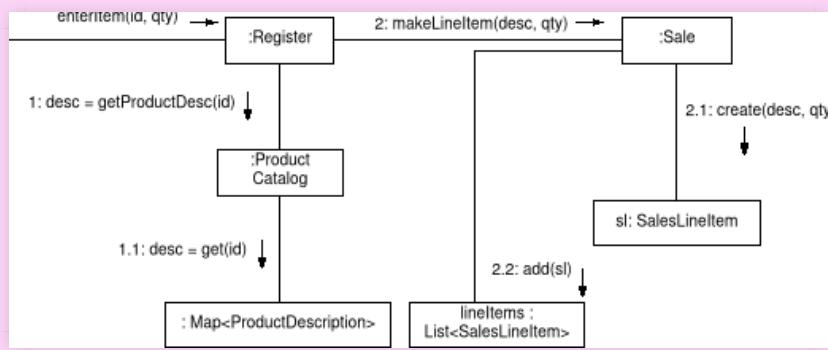
```
LogManager.getInstance().log( "aaaaaaaah haaaaahhaa");
```

CRÉATIVITÉ ET CHANGEMENTS PENDANT LE CODAGE

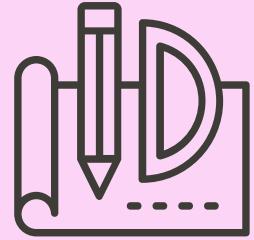


- Les artefacts de conception affectent l'implémentation
- Pendant la programmation et les tests
 - multitudes de changements peuvent surgir
- S'attendre à des déviations et les prévoir
 - Favoriser le changement
- Le codage n'est pas une activité triviale
 - Pourquoi?

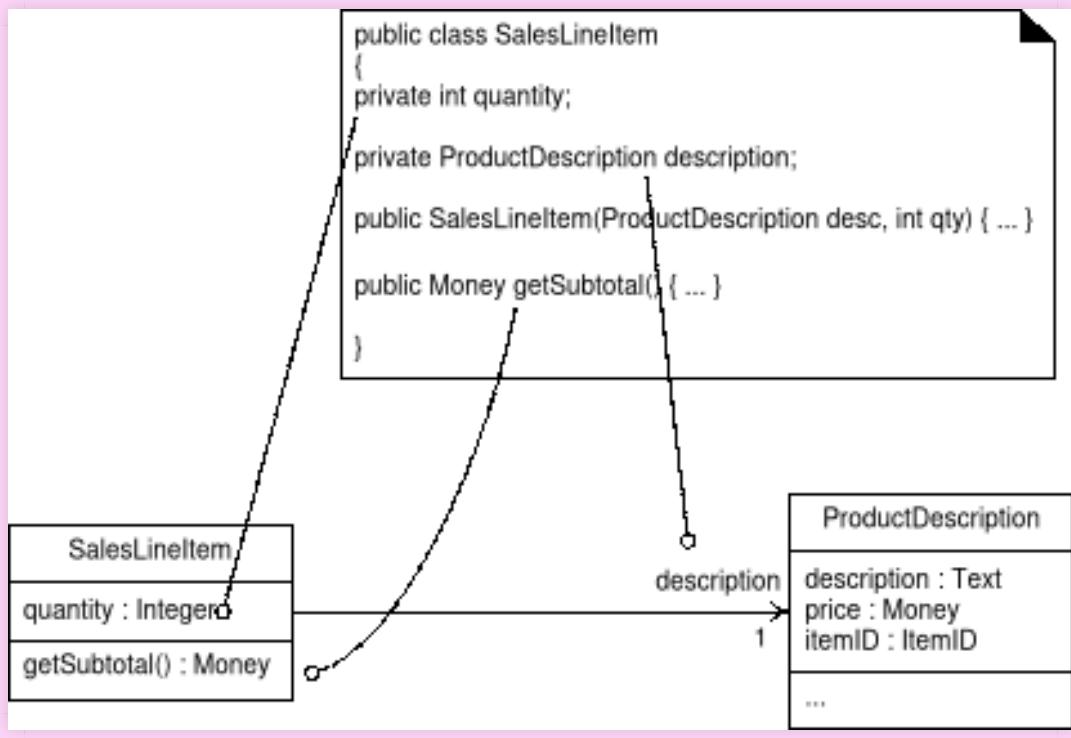
MÉTHODES À PARTIR DES DIAGRAMMES D'INTERACTION



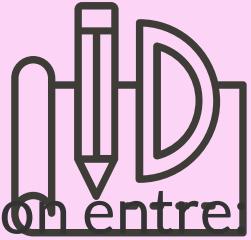
SALESLINEITEM



- SalesLineItem en Java

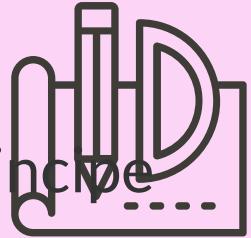


RÉSUMÉ



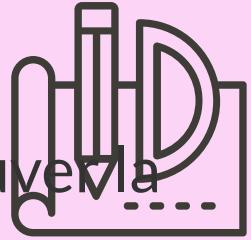
- Nous avons vu des processus de traduction entre
 - les diagrammes de classes UML et les définitions de classes
 - les diagrammes d'interaction UML et les corps des méthodes
- Mais le travail de programmation laisse encore beaucoup de place à l'exploration

VALIDATION DE LA COMPRÉHENSION



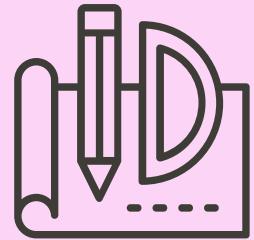
- Quelle est la motivation principale du principe GRASP Créateur?
 1. Pour avoir une création rapide (performante) d'instances
 2. Pour obtenir une conception plus facile à comprendre
 3. Pour avoir une conception qui prend moins de mémoire
 4. Pour identifier des occasions d'appliquer le patron Fabrique

VALIDATION DE LA COMPRÉHENSION



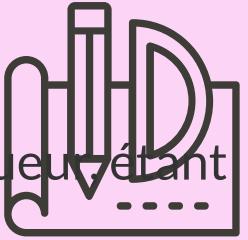
- Le pattern Contrôleur GRASP sert à trouver la bonne classe...
 1. de la couche présentation pour réaliser une opération système
 2. de la couche domaine pour réaliser une opération système
 3. de l'IHM pour implémenter une logique applicative
 4. de l'IHM pour contrôler le système

VALIDATION DE LA COMPRÉHENSION

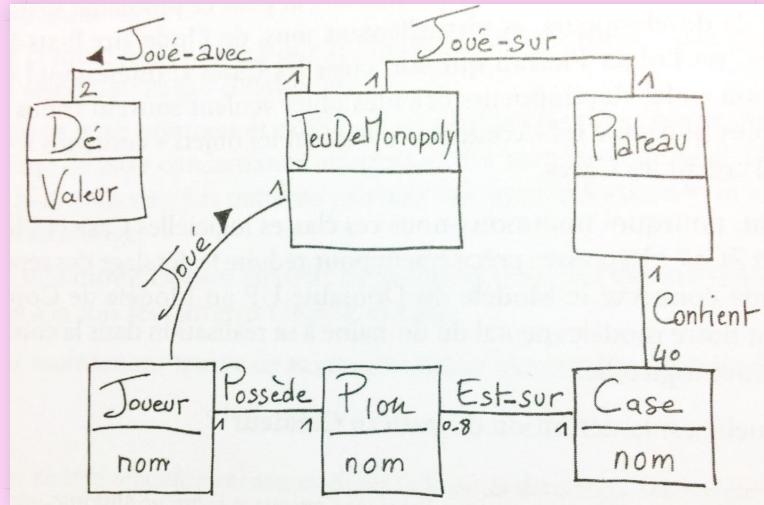


- Quel genre de contrôleur GRASP est JeuDeMonopoly dans l'exemple?
 1. Contrôleur de jeu
 2. Contrôleur MVC
 3. Contrôleur de session
 4. Contrôleur de façade

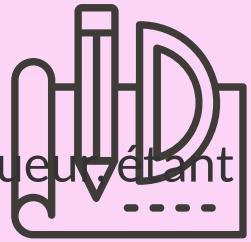
VALIDATION DE LA COMPRÉHENSION



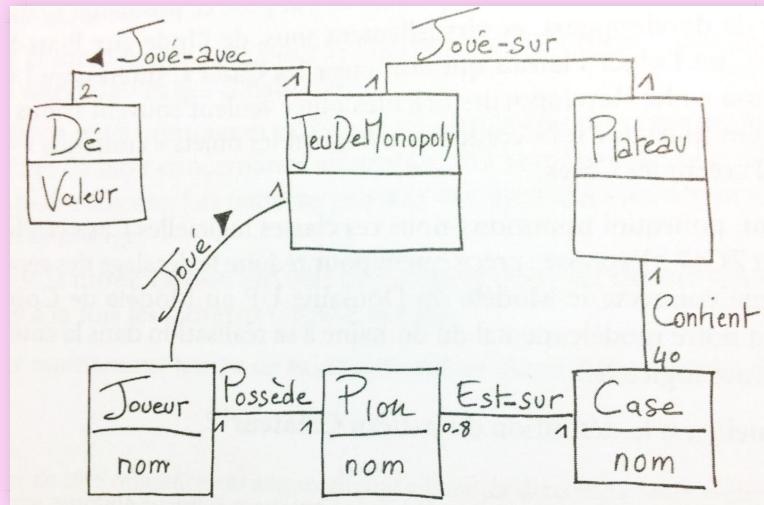
- Selon la figure A17.3/F16.3, qui connaît un objet Joueu[er] étant donné le nom du joueur?



VALIDATION DE LA COMPRÉHENSION

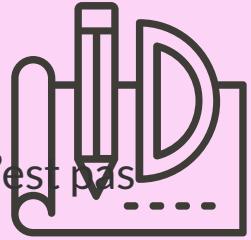


- Selon la figure A17.3/F16.3, qui connaît un objet Joueur étant donné le nom du joueur?



Quel est ce patron GRASP?

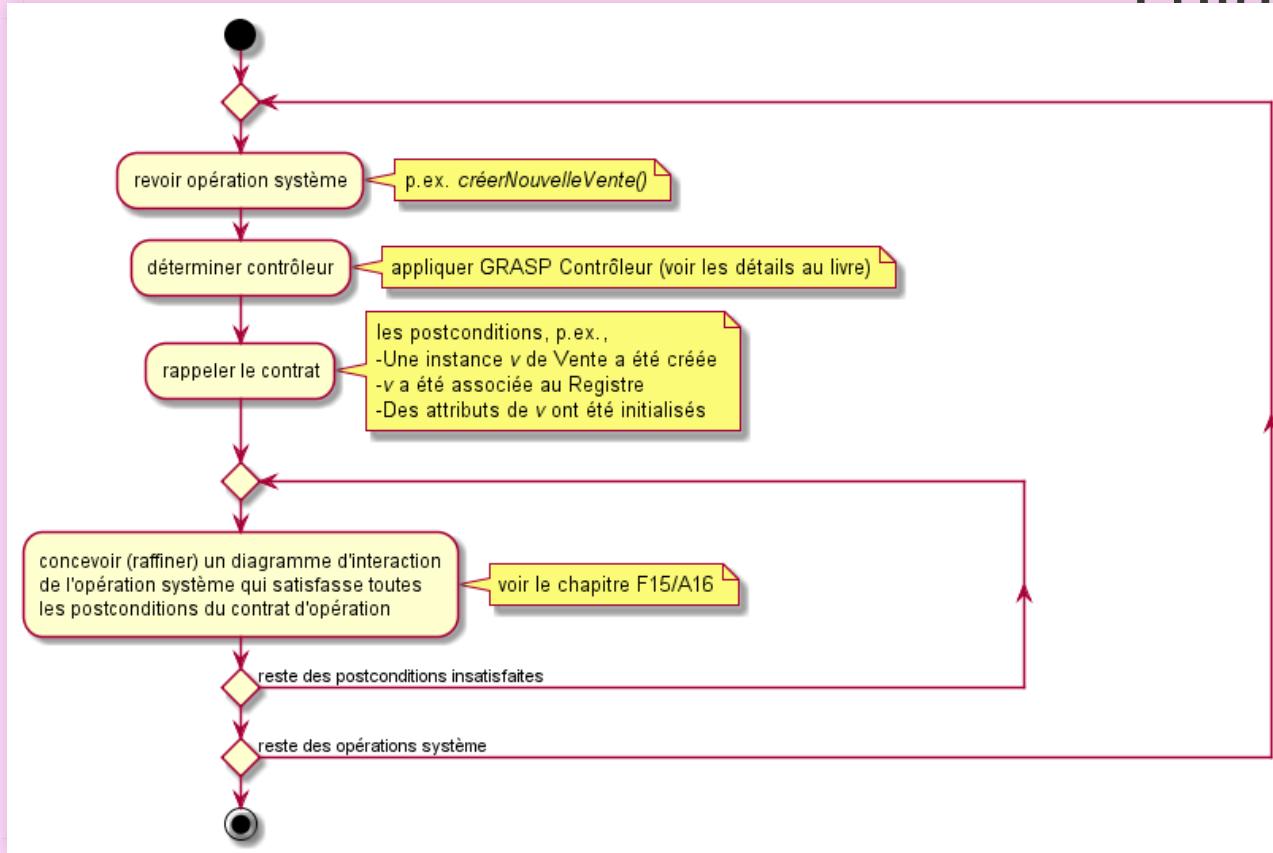
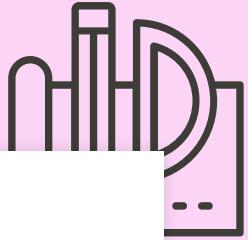
RÉALISATION DE CAS D'UTILISATION NEXTGEN



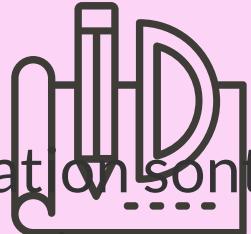
- La zone non-magie
 - Création de diagrammes d'interaction bien conçus n'est pas magique!
 - Tout s'appuie sur des principes de bonne conception justifiables.
 - GRASP (General Responsibility Assignment Software Pattern)
- L'examen final est aussi une zone non-magie!
 - Il faut y faire preuve d'utilisation des principes justifiables (GRASP)



PROCESSUS DE HAUT NIVEAU



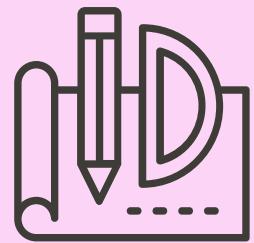
RÉALISATION DE CAS D'UTILISATION



- Cela « décrit la façon dont les cas d'utilisation sont réalisés dans le Modèle de Conception, en termes d'objets qui collaborent » [RUP]
 - Réalisation de scénario
- Cela fait le lien entre exigences et conception
- C'est un diagramme d'interaction
- C'est une documentation des décisions de COO
 - annotations GRASP

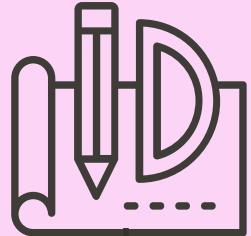
CHP17 CONCEPTION AVEC LES GRASP

1. Qu'est-ce qu'une RDCU?



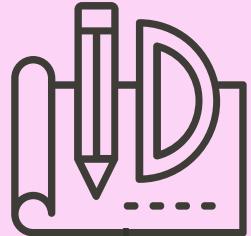
13 . 66

CHP17 CONCEPTION AVEC LES GRASP



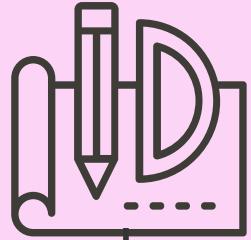
1. Qu'est-ce qu'une RDCU?
2. Quelle est l'utilité des contrats d'opérations dans une RDCU?

CHP17 CONCEPTION AVEC LES GRASP



1. Qu'est-ce qu'une RDCU?
2. Quelle est l'utilité des contrats d'opérations dans une RDCU?
3. Est-ce que le retour d'information dans une opération du DSS est important pour la création d'une RDCU?

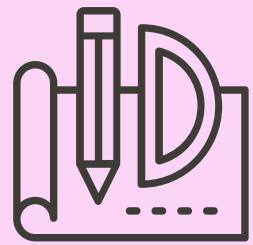
CHP17 CONCEPTION AVEC LES GRASP



1. Qu'est-ce qu'une RDCU?
2. Quelle est l'utilité des contrats d'opérations dans une RDCU?
3. Est-ce que le retour d'information dans une opération du DSS est important pour la création d'une RDCU?
 - Pourquoi?

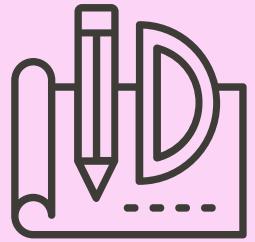
CHP18 VISIBILITÉ

1. Nommez les 4 types de visibilités.



13 . 67

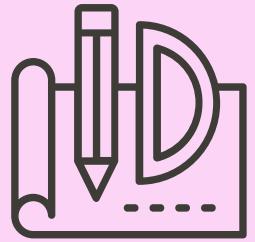
CHP18 VISIBILITÉ



1. Nommez les 4 types de visibilités.
2. Comment déterminer l'ordre d'implémentation des classes?

13 . 67

CHP18 VISIBILITÉ

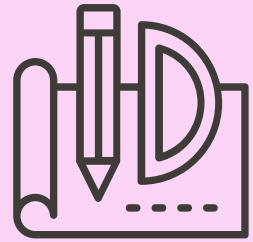


1. Nommez les 4 types de visibilités.
2. Comment déterminer l'ordre d'implémentation des classes?
 - Pourquoi?

13 . 67

EXERCICE RDCU

Correction

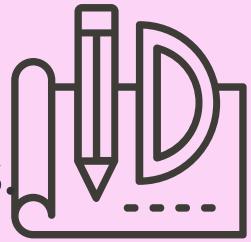


<https://github.com/yvanross/LOG210-exercices>

13 . 68

RDCU - AIDE MÉMOIRE

Voir la figure 8.1 des notes de cours.

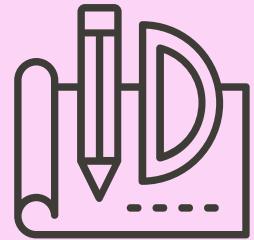


AUTRES PATTERNS D'AFFECTION DES RESPONSABILITÉS



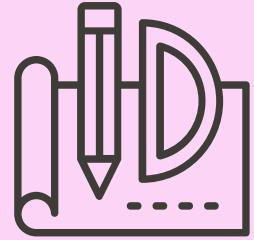
13 . 70

LES NEUF PATRONS GRASP



- Expert
- Créeateur
- Contrôleur
- Faible Couplage
- Forte Cohésion
- Polymorphisme
- Fabrication pure
- Indirection
- Protection des variations

POLYMORPHISME



- Principe de base pour la conception OO
- Cas qui varient mais qui sont similaires
- Dans une famille de classes
 - opération polymorphique
 - dans laquelle les cas sont différents
- Formes géométriques (carré, cercle, triangle, etc.)
 - se dessine avec une méthode draw()
 - chaque classe implémente sa méthode selon son cas...

POLYMORPHISME

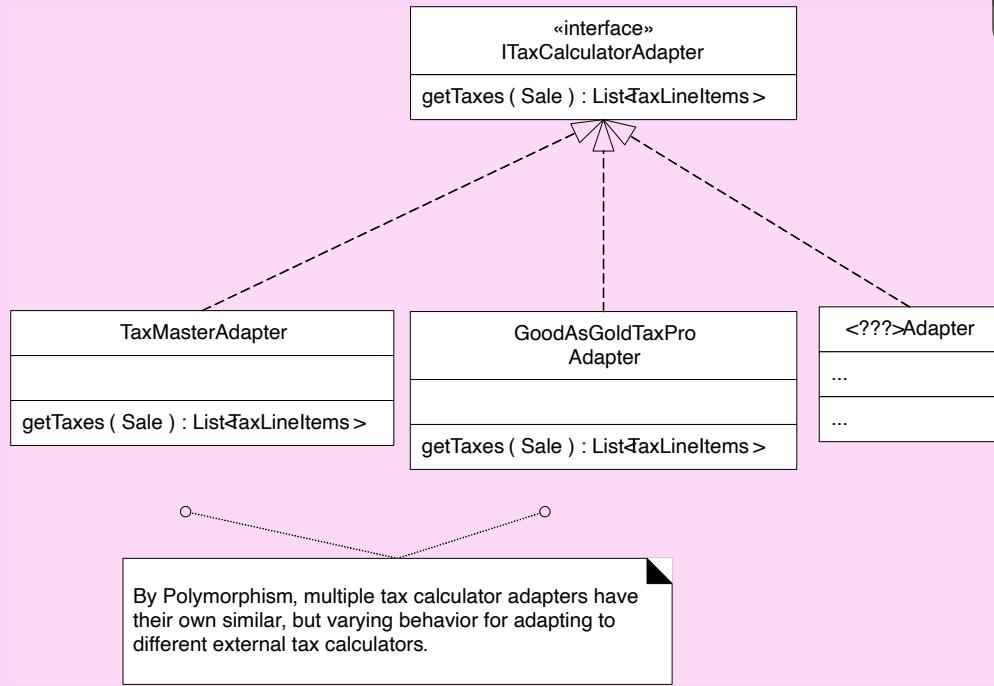
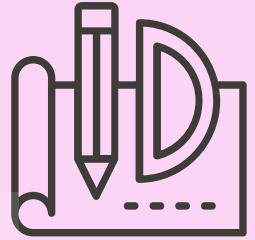


Fig. 25.1 (3e édition)

POLYMORPHISME

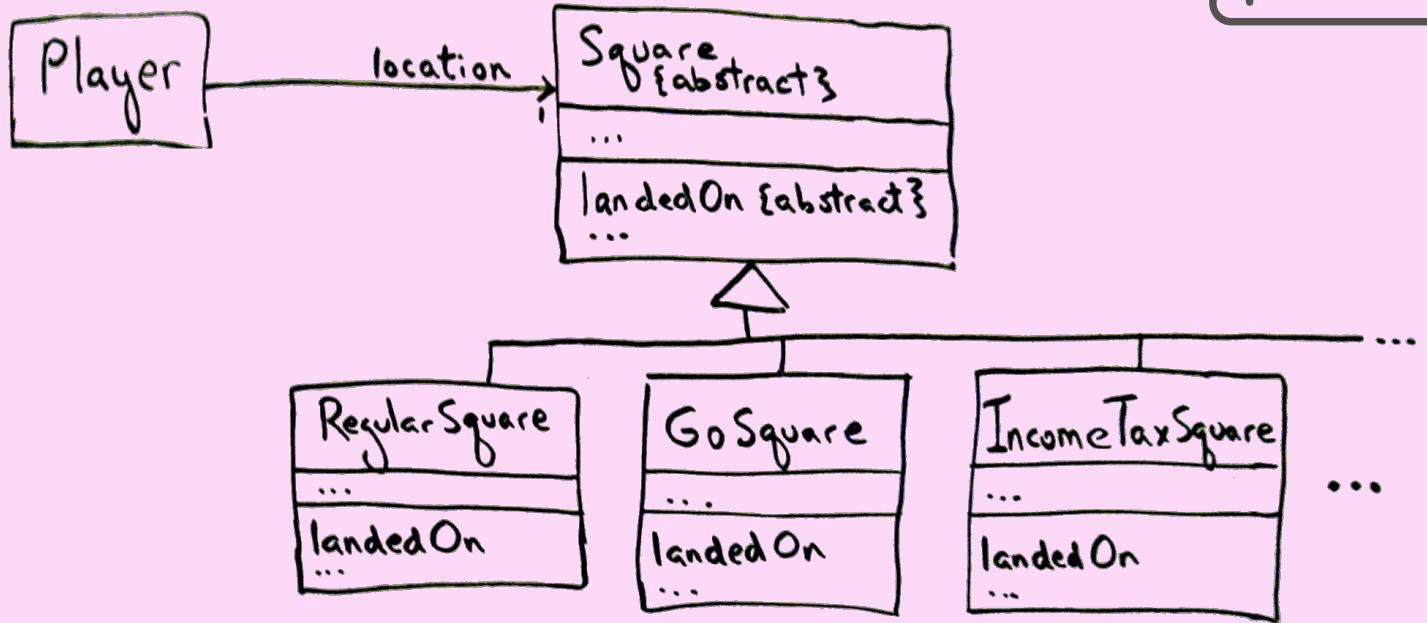
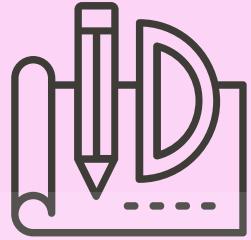


Fig. 25.2 (3e édition)

POLYMORPHISME

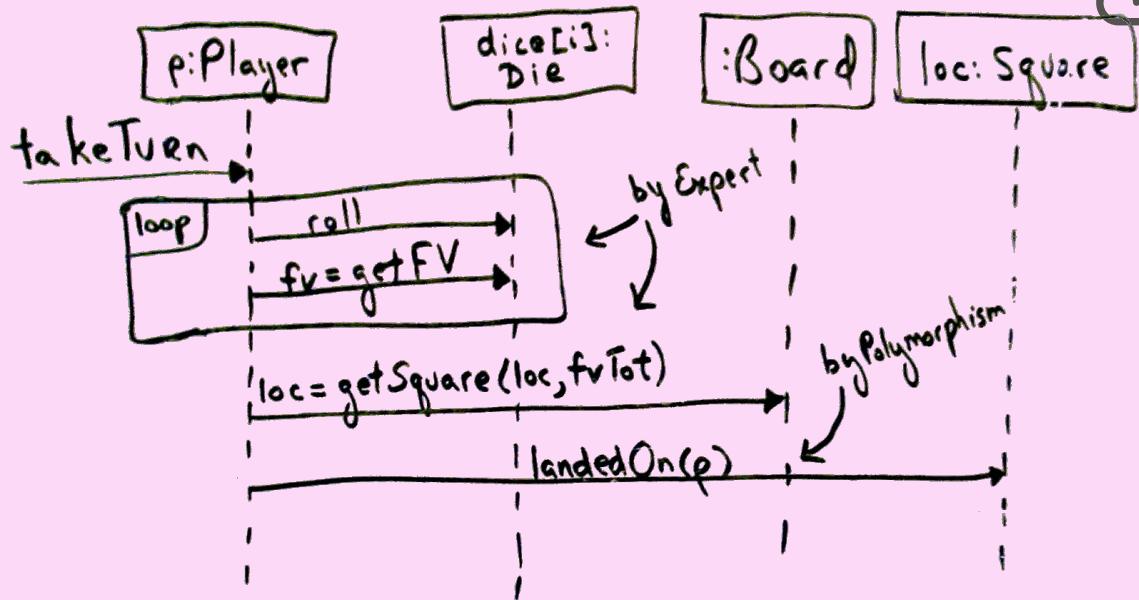
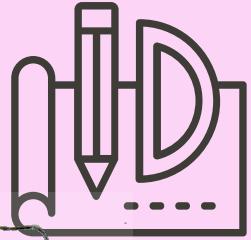


Fig. 25.3 (3e édition)

POLYMORPHISME

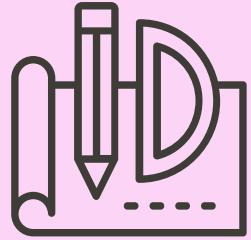
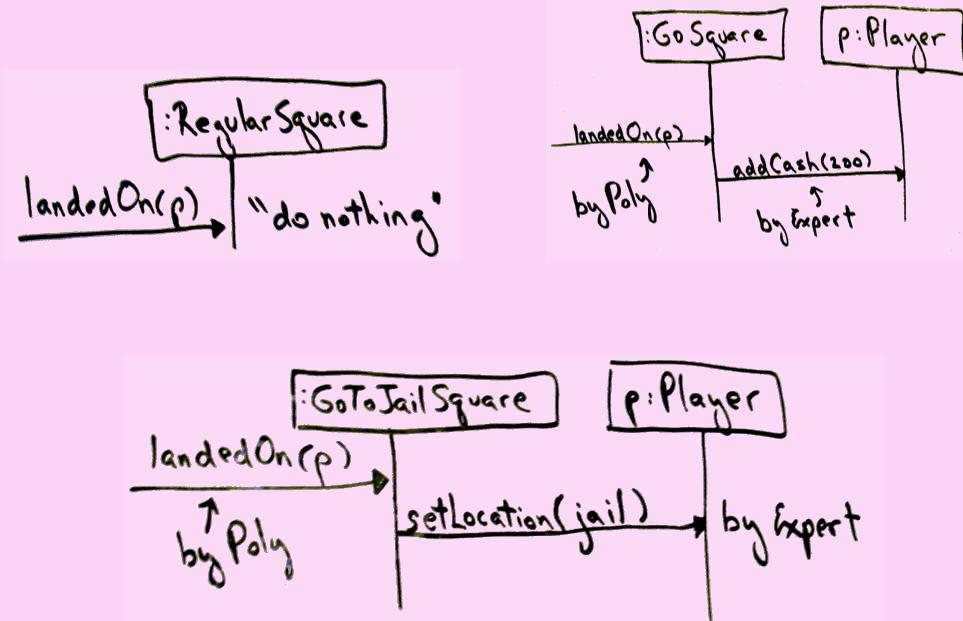
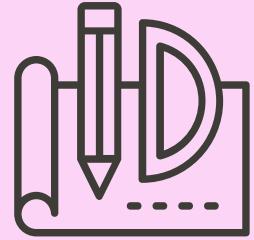


Fig. 25.4, 25.5, 25.7 (3e édition)



FABRICATION PURE



- Parfois en suivant un patron
 - assigne une responsabilité
 - mais il y a des conséquences non désirables
 - augmentation du couplage
 - diminution de la cohésion
- Solution
 - fabrique une classe « comportementale »
 - qui n'a pas d'équivalent dans le modèle du domaine
 - invention de la part du concepteur

FABRICATION PURE

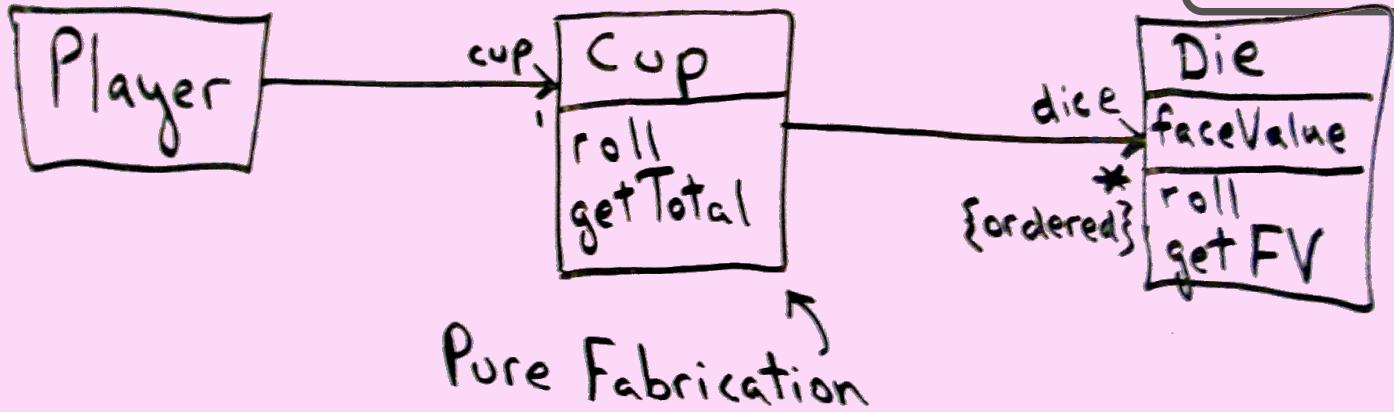
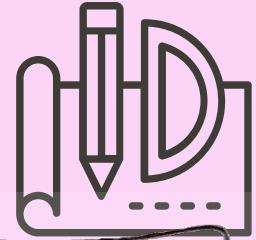


Fig. 25.8 (3e édition)

FABRICATION PURE

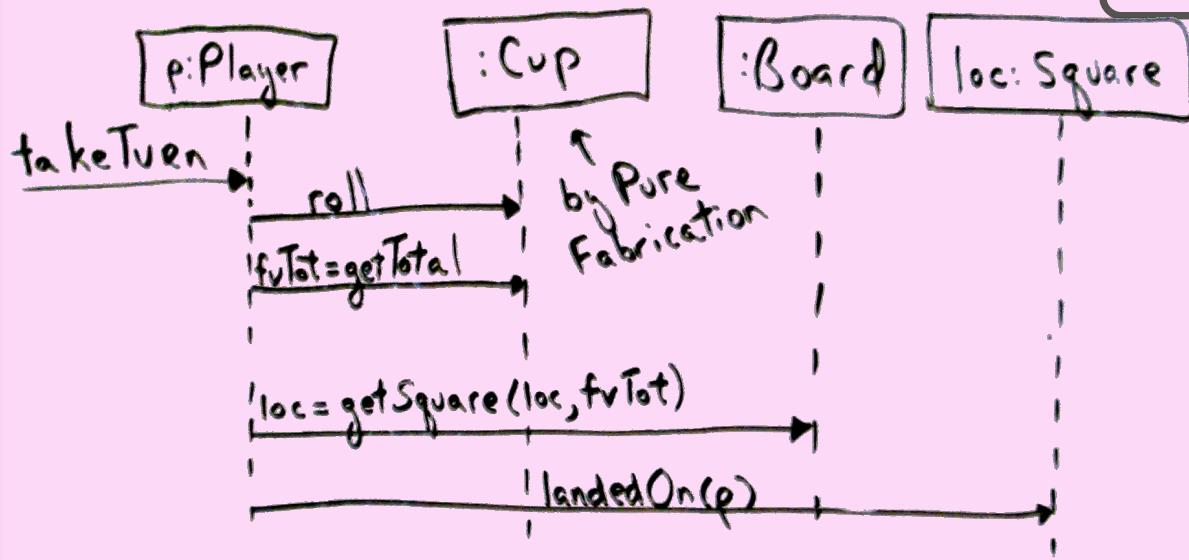
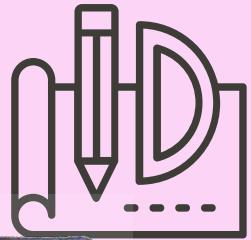


Fig. 25.9 (3e édition)

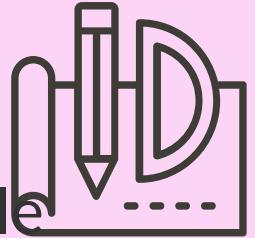
COMPRÉHENSION



Quel autre GRASP implique une fabrication pure? -----
Autrement dit, on propose une classe « inventée » ne faisant pas partie du MDD.

1. Créateur
2. Contrôleur
3. Expert
4. Forte cohésion
5. Faible couplage

INDIRECTION



- Mécanisme souvent utilisé pour réduire le couplage
- Objet intermédiaire
 - découple deux composants

INDIRECTION

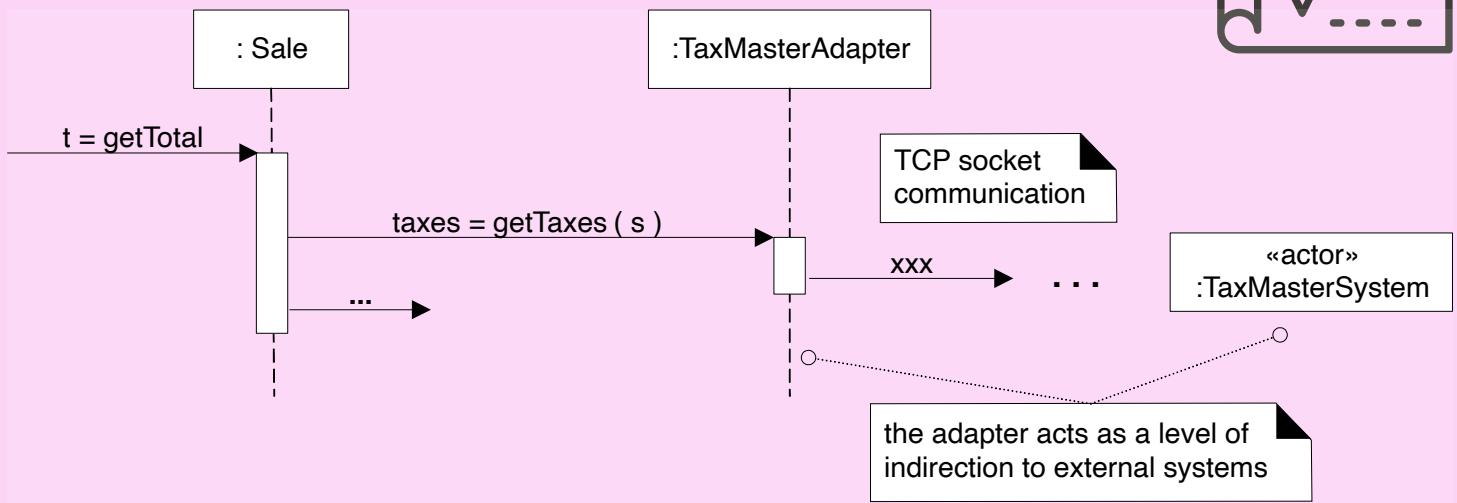
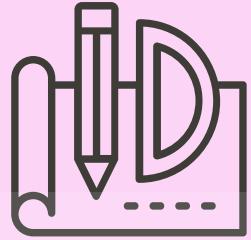
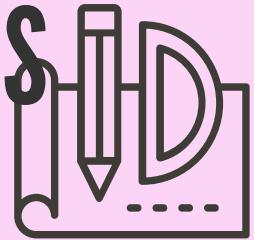


Fig. 25.10 (3e édition)

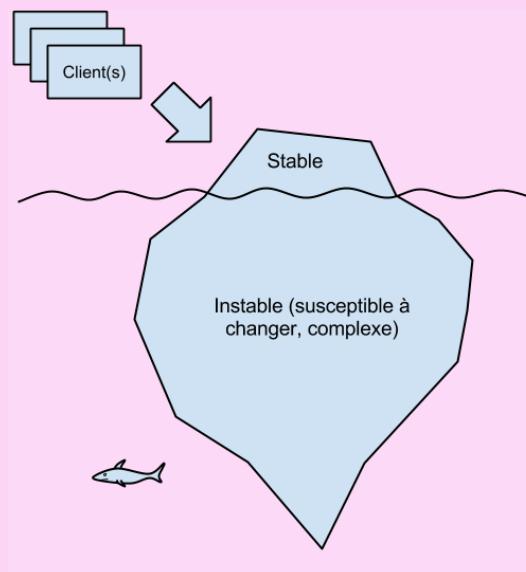
PROTECTION DES VARIATIONS



- Composants (classes, interfaces, ...)
 - variations (ou instabilités) de ces composants en affectent d'autres
- Éliminer ces effets indésirables
- Identifier des zones de variabilité

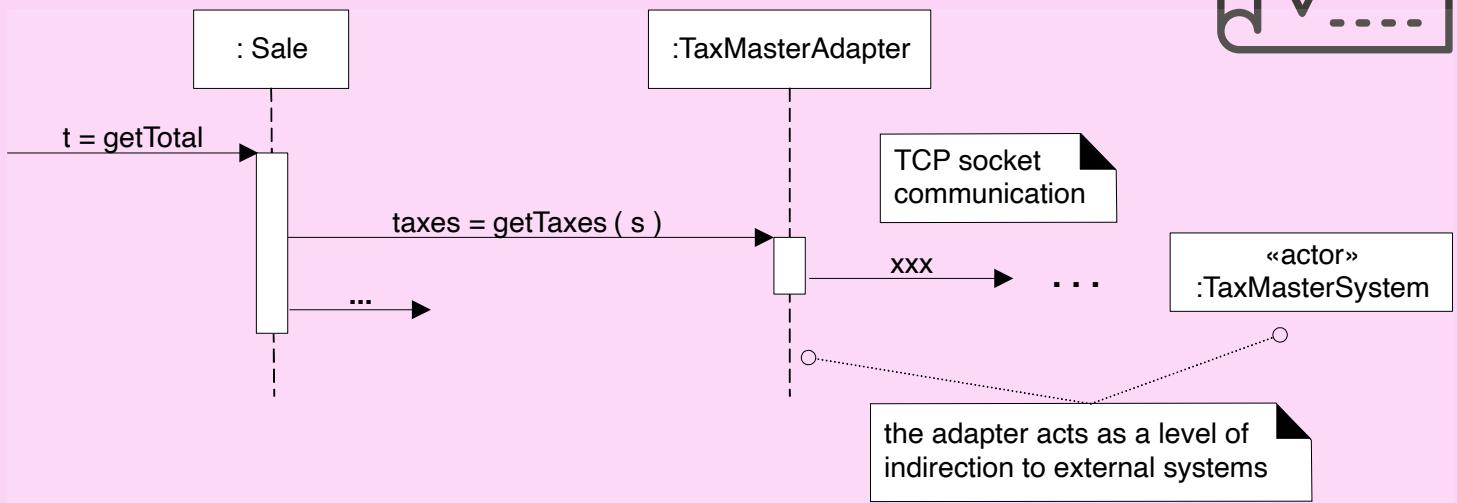
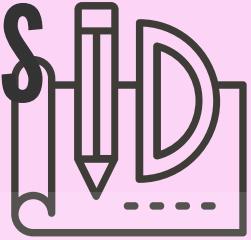
PROTECTION DES VARIATIONS

- Créer une interface stable (API) autour des composants instables

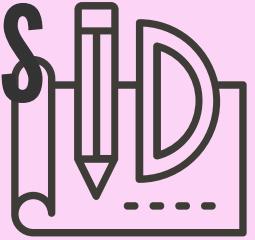


13 . 84

PROTECTION DES VARIATIONS

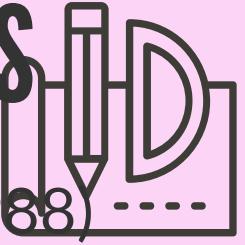


PROTECTION DES VARIATIONS



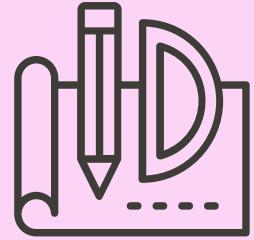
- Vieux principe du génie logiciel
- Principe de « Information hiding » (Parnas, 1972)
 - Encapsulation des données
 - Déterminer
 - choix difficiles du design
 - choix du design qui auront tendance à changer
 - Concevoir chaque module afin de cacher ces choix « variables »

PROTECTION DES VARIATIONS



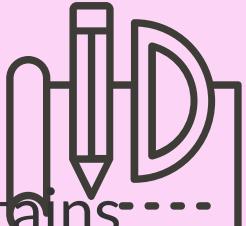
- Principe de « Open-closed » (Meyer, 1988) ----
- Composants devraient être
 - ouverts à l'extensibilité (Implémentations)
 - fermés à la modification (API)
- Ces contraintes paraissent contradictoires

RÉSUMÉ



- D'autres patrons GRASP
 - Polymorphisme
 - Fabrication pure
 - Indirection
 - Protection des variations
- Vocabulaire
 - concepts de base pour les ingénieurs en logiciel
 - améliore la communication dans une équipe

GRASP DANS LES GOF



- Adaptateur est une spécialisation de certains principes GRASP

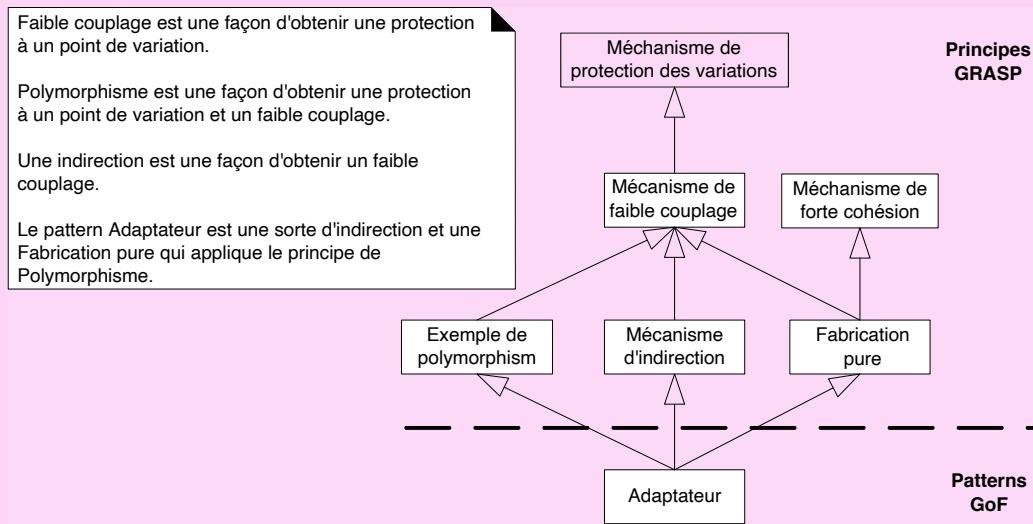
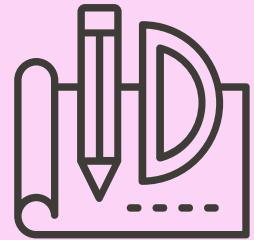


fig. F23.3

GRASP DANS LES GOF



Exercice Google Classrooms

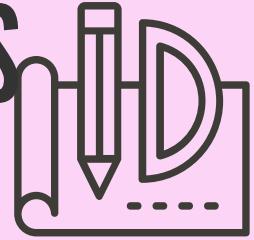
“Identification des GRASP dans les GoF”

Salles de travail Zoom

20 minutes, suivre le modèle

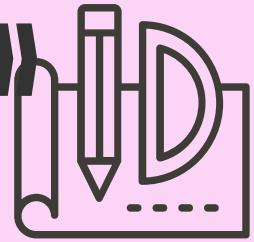
13 . 90

TRAITEMENT DE PAIEMENTS



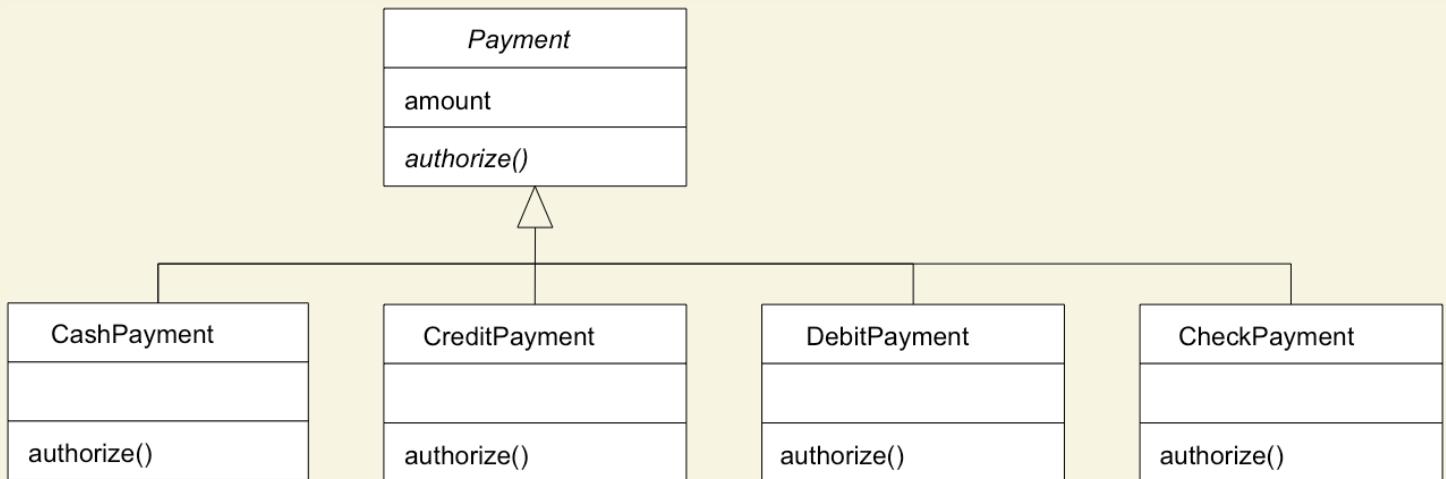
- Multiple paiements
 - chèque
 - carte de crédit
 - carte de débit
 - comptant

PATRON « FAIRE SOI-MÊME »



- Objet logiciel
 - abstraction d'un objet réel
 - fait ce qui est normalement fait à l'objet réel
 - avantage: réduire le décalage dés représentations
- Exemple
 - objet Texte qui vérifie sa propre orthographe

PAIEMENT ET POLYMORPHISME



By Polymorphism, each payment type should authorize itself.

This is also in the spirit of "Do it Myself" (Coad)

fig. F33.17

PAIEMENT PAR CHÈQUE

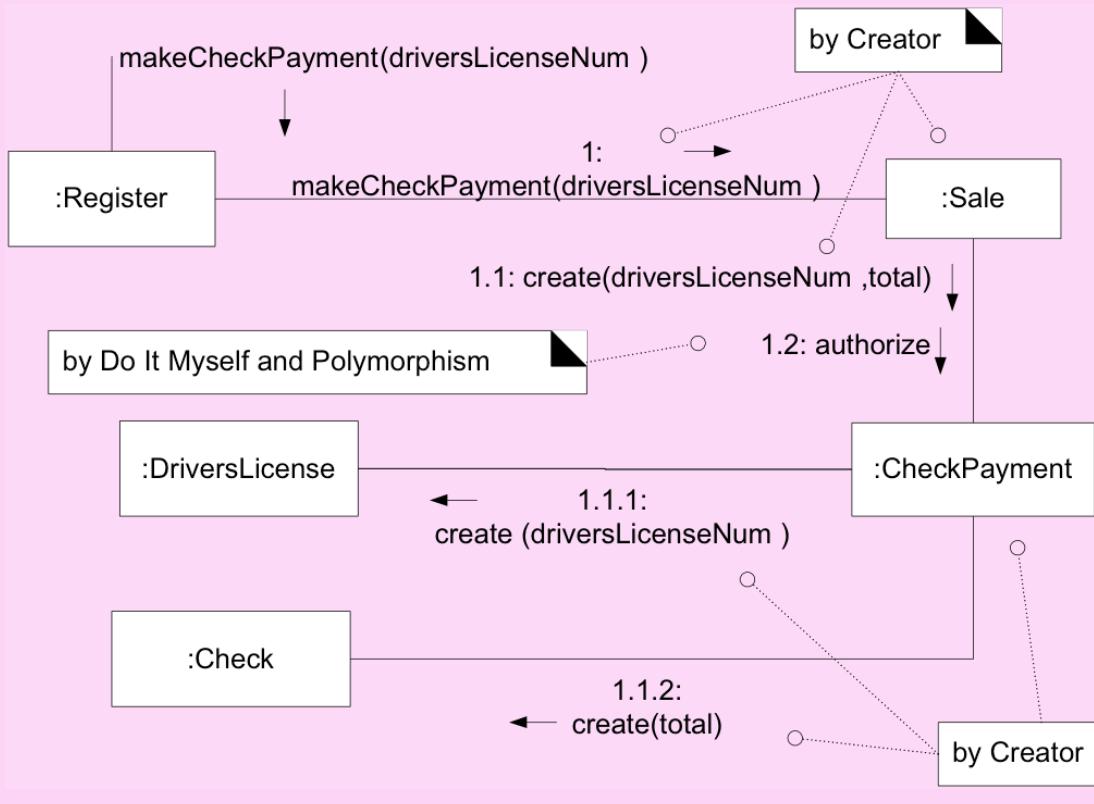
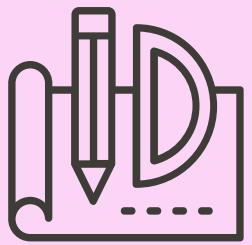


fig. F30.19

PAIEMENT PAR CARTE DE CRÉDIT

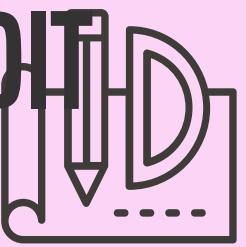
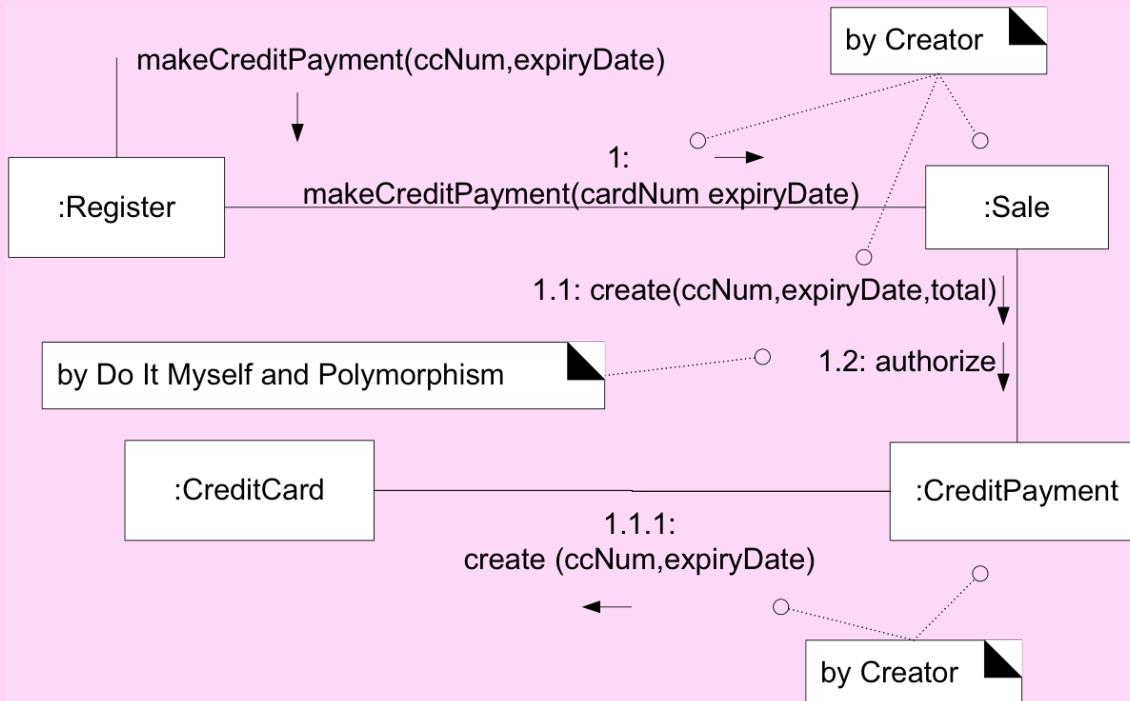



fig. F30.18

AUTORISATION

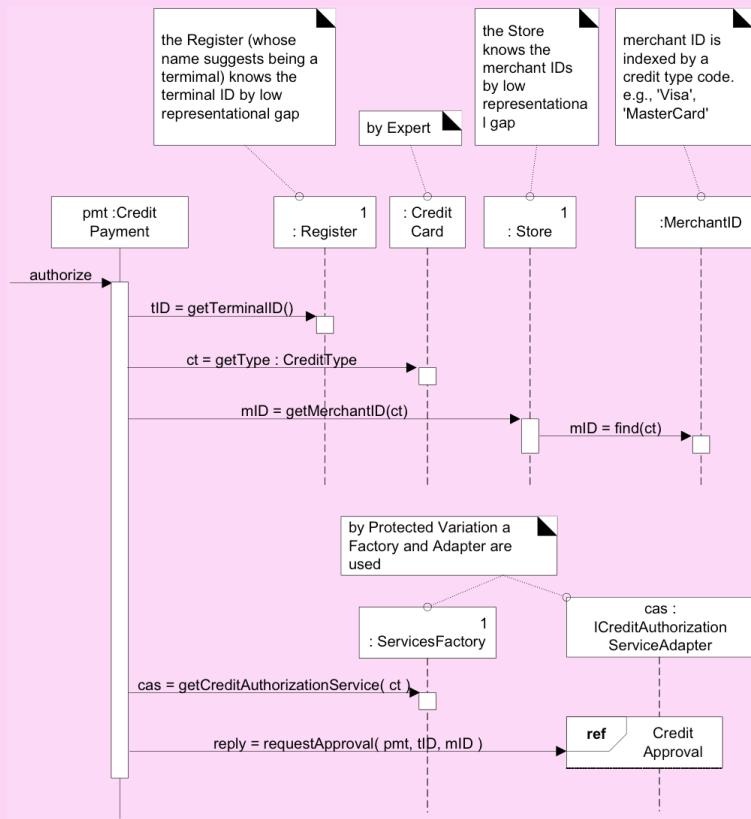
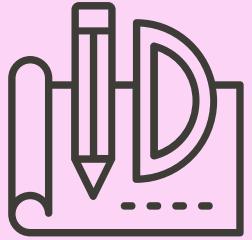


fig. F30.20

AUTORISATION

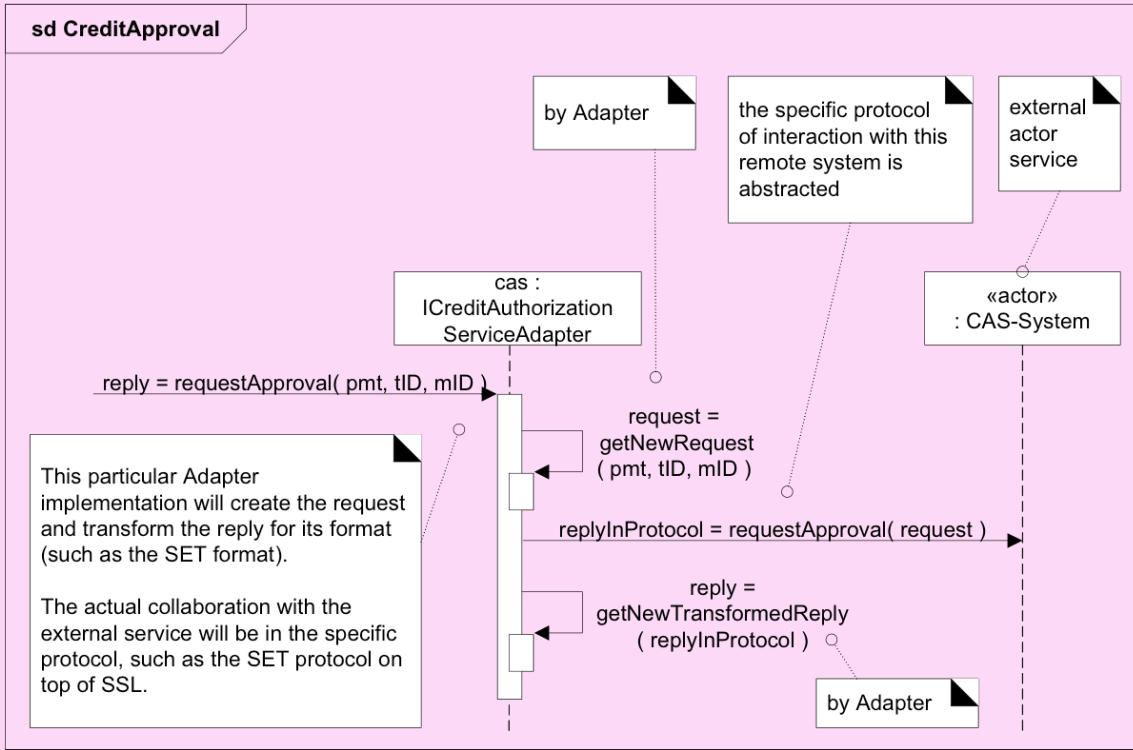
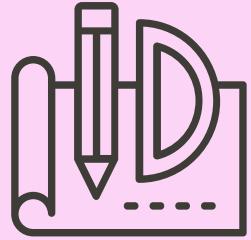


fig. F30.21

PAIEMENT APPROUVÉ

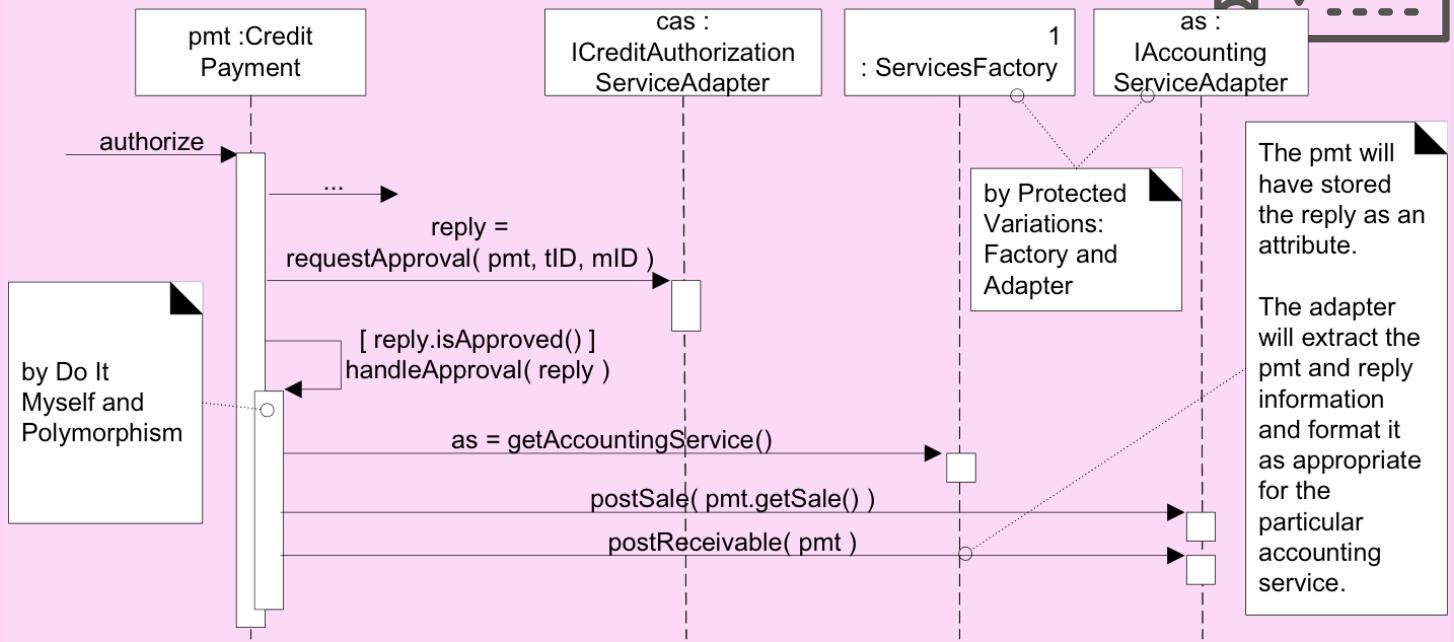
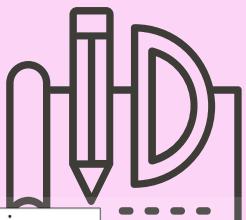
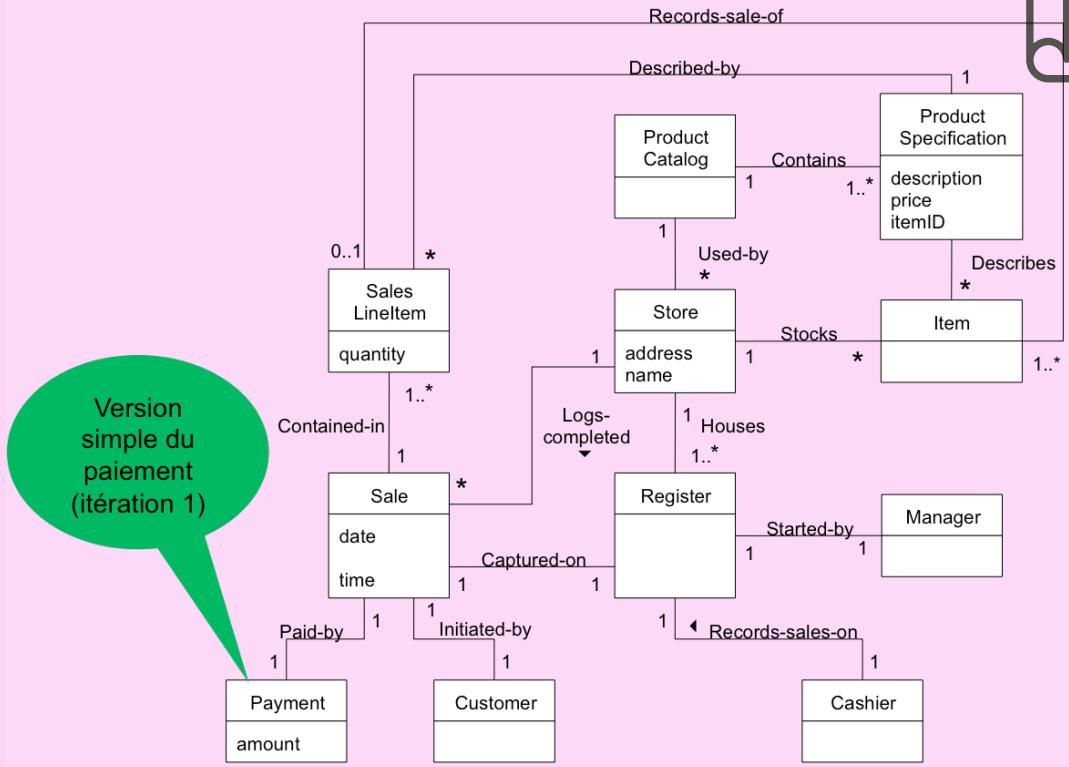
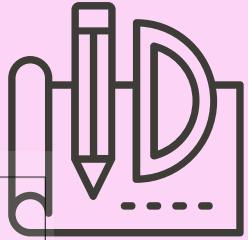
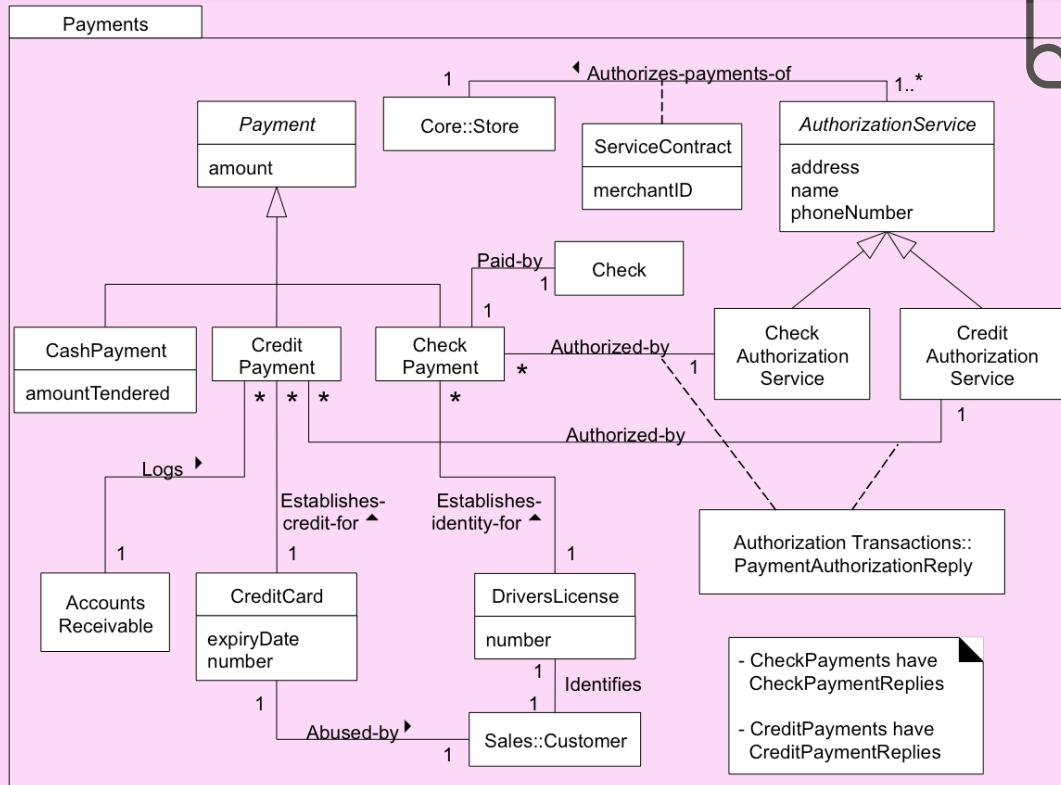
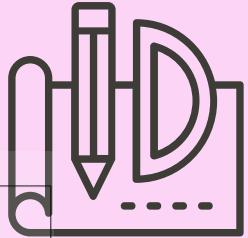


fig. F30.22

VERSION SIMPLE DU PAIEMENT



PACKAGE: PAIEMENT



MODULE DIAGRAMME DE CLASSE LOGICIELLES

DCL

1. Diagramme de classe logiciel- 08.59m
2. Attributs - 15.11m
3. Révision

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

DCL - DCC

DIAGRAMME DE CLASSE LOGICIEL



Created by Swen-Peter Ekkebus
from the Noun Project

DIAGRAMME DE CLASSE DE CONCEPTION

14 . 2

CLASSIFICATEURS UML



Created by Swen-Peter Ekkebus
from the Noun Project

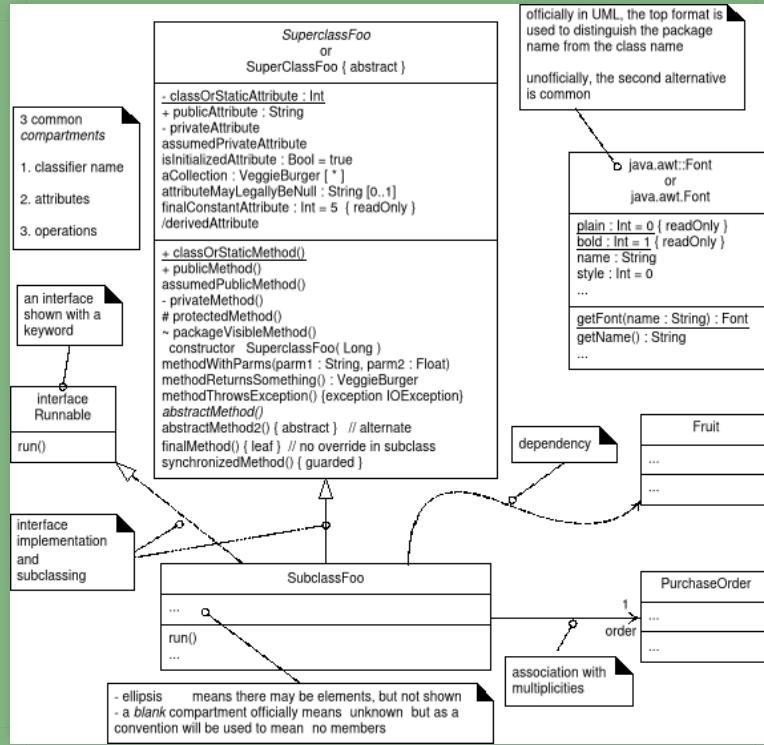
- Dans les DCC, il y en a deux classificateurs courants:
 - Les classes
 - Les interfaces

<https://plantuml.com/class-diagram>

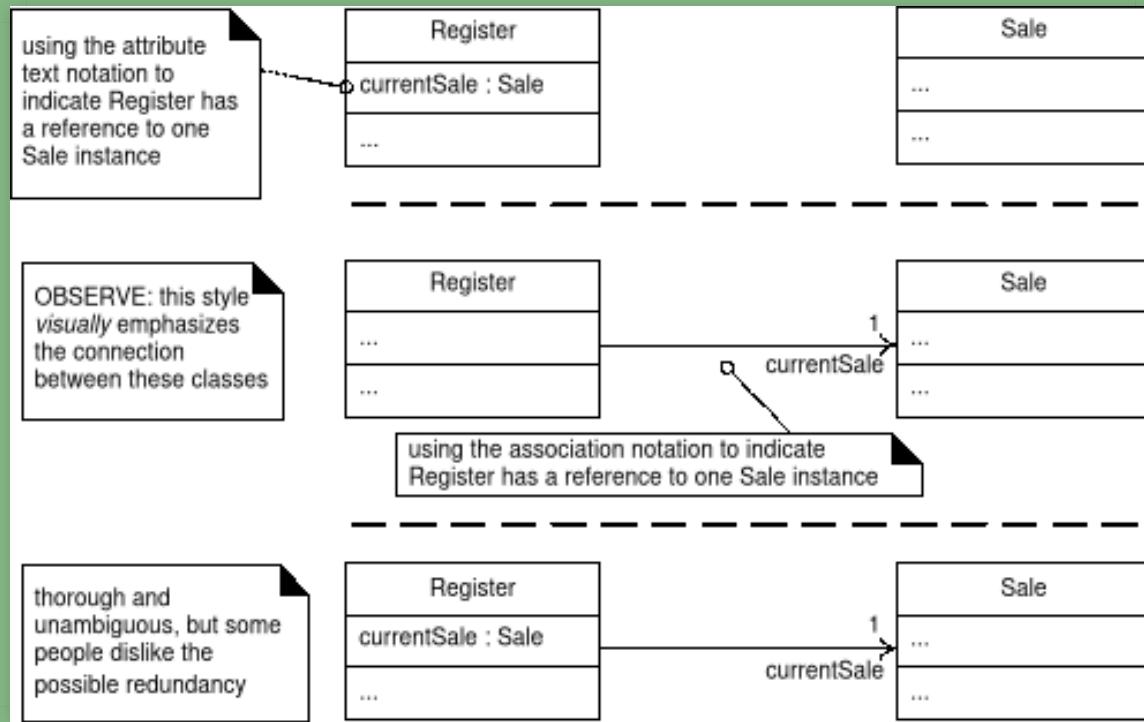
NOTATION COURANTE (VUE EN LOG2A)



Created by Sven-Peter Ekkebus
from the Noun Project



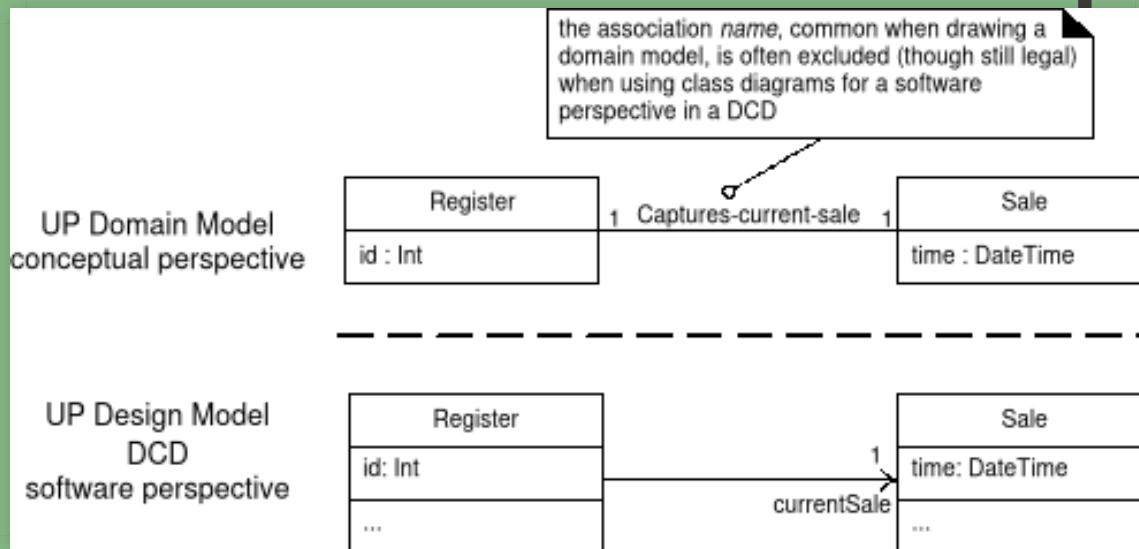
ATTRIBUTS EN UML



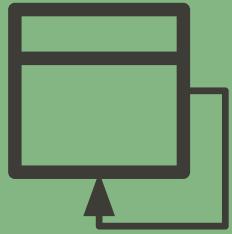
DÉCALAGE DE PRÉSENTATION



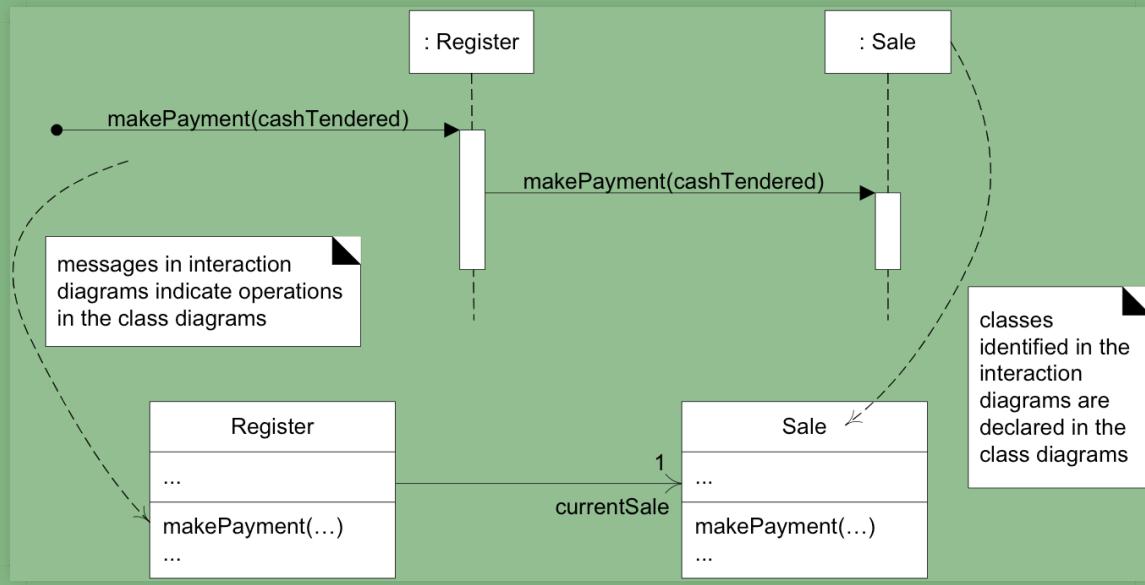
©n-Peter Ekkebus
Project



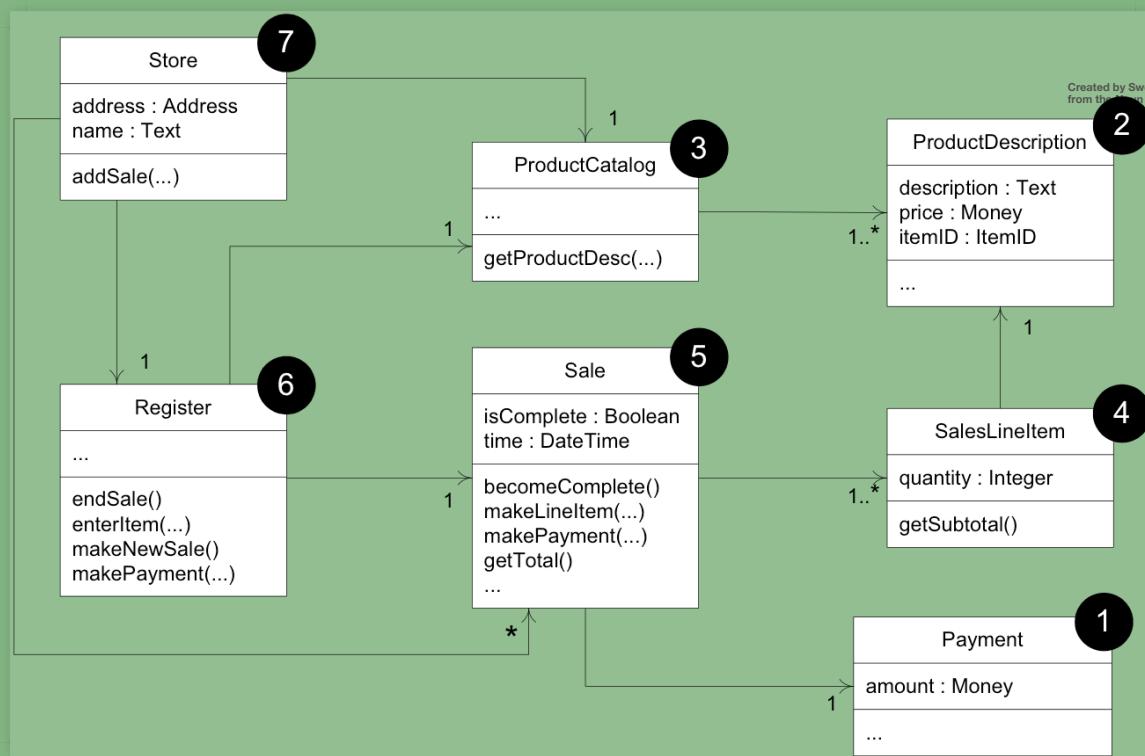
RELATION DIAGRAMMES D'INTERACTION ET DE CLASSES



Created by Swen-Peter Ekkebus
from the Noun Project



ORDRE D'IMPLEMENTATION



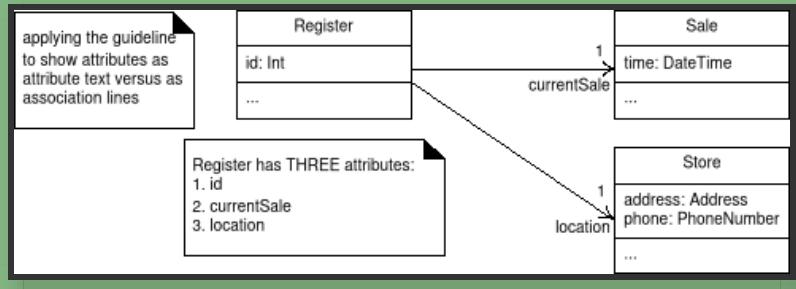
RÉSUMÉ



Created by Swen-Peter Ekkebus
from the Noun Project

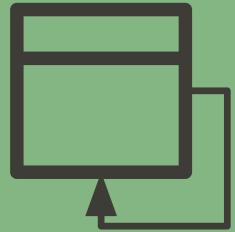
- DCC servent comme plan pour le codage
 - Soyez à l'aise avec la transformation
- DCC se construisent au fur et à mesure que l'on réalise la dynamique (à travers les diagrammes d'interaction)

DEUX FAÇONS DE NOTER LES ATTRIBUTS



```
public class Register {  
    private int id;  
    private Sale currentSale;  
    private Store location;  
    // ...  
}
```

CLASSES COLLECTION

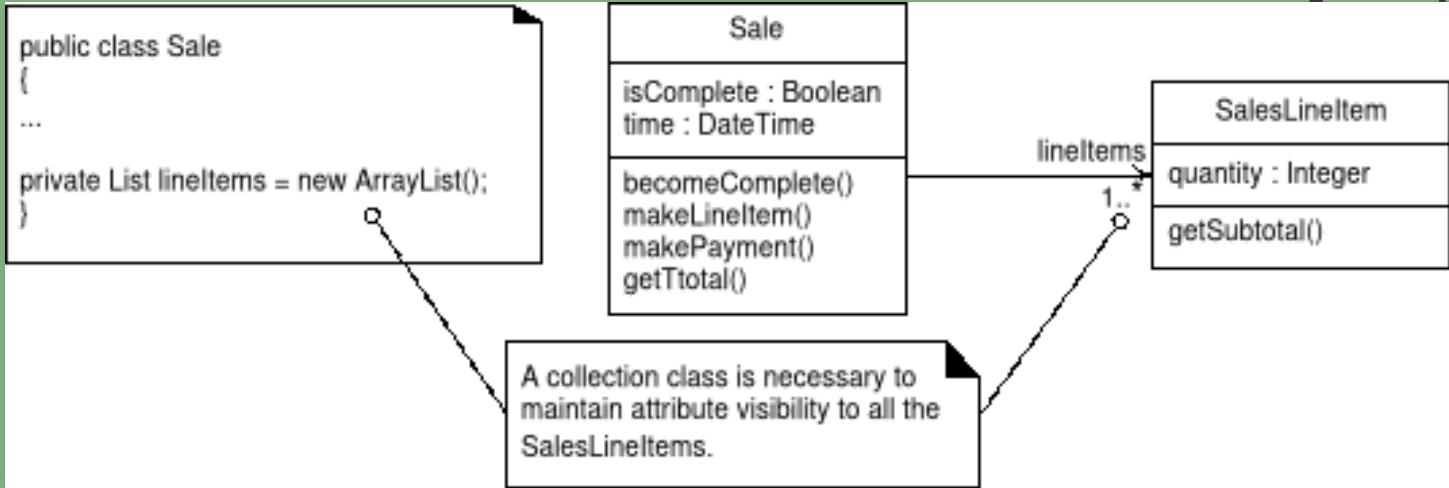


Created by Swen-Peter Ekkebus
from the Noun Project

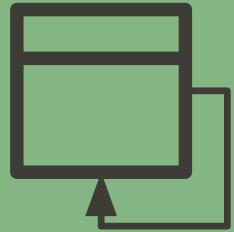
- Multiplicités des associations
- Besoin d'un groupe d'objets similaires
- Choix du type de collection
 - influencé par le besoin

```
public class Sale{  
    private List<SalesLineItem> lineItems =  
        new ArrayList<SalesLineItems>();  
    // ...  
}
```

AJOUTER UNE CLASSE DE COLLECTION



OPÉRATION VS MÉTHODE



Created by Swen-Peter Ekkebus
from the Noun Project

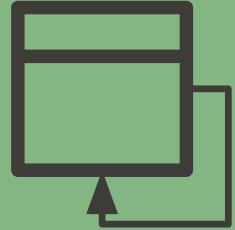
- UML DCC : signature des opérations
 - Déclaration (décision de conception)
 - Opérations sont quasiment « vides » en UML
- Java : implémentation des méthodes
 - Implémentation (décision de codage)
- Déclaration d'**opération** (UML) est différente de l'implémentation de **méthode** (langage OO)
 - P.ex. un contrat d'opération définit les contraintes sur une opération et non une méthode
 - La différence est subtile mais importante

MOTS CLÉS, STÉRÉOTYPES, PROFILS, ÉTIQUETTES

Created by Swen-Peter Ekkebus
from the Noun Project

- Voir les sections A16.7/F15.7, A16.8/F15.8
 - Stéréotype <create> , <destroy>, <metaclass>, <Actor>, <Interface>, ...
 - Le stéréotype déclare un ensemble d'étiquettes
- Etiquette: @login_required
 - Fonctionnalités et attributs que l'on veux donner à une classe ou une méthode.

NOTATION DE DÉPENDANCE



Created by Swen-Peter Ekkebus
from the Noun Project

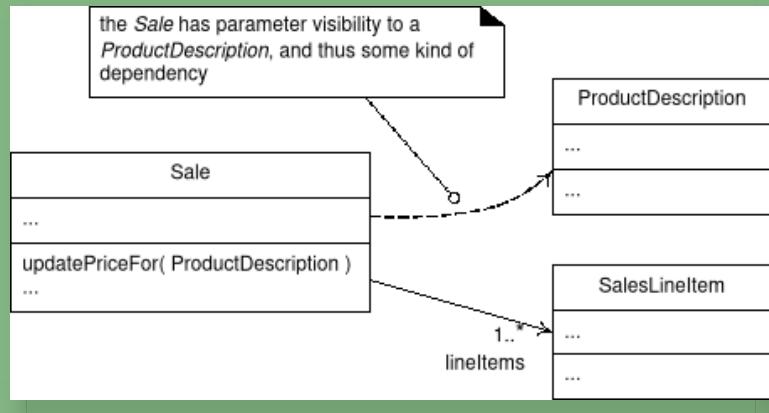
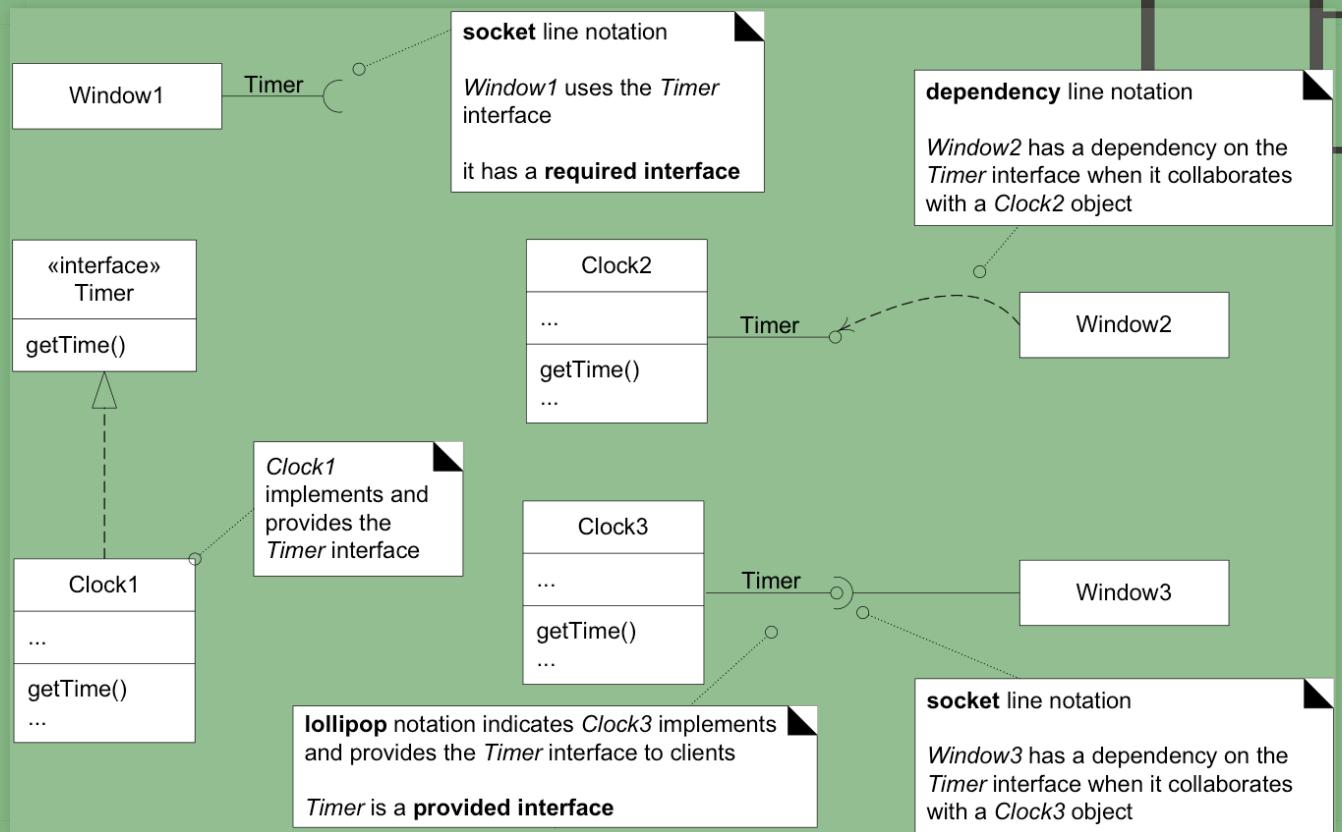


fig F15.9

```
public class Sale{
    public void updatePriceFor(
        ProductDescription
        description) {
            Money basePrice = description.getPrice();
            // ...
        }
        // ...
}
```

• Interfaces et réalisation d'interface



CHP15 DIAGRAMME DE CLASSE

1. Quelle est la différence entre un diagramme de classes conceptuelles et logicielles?

Created by Swen-Peter Ekkebus
from the Noun Project

14 . 17

CHP15 DIAGRAMME DE CLASSE



1. Quelle est la différence entre un diagramme de classes conceptuelles et logicielles?
2. Quelles sont les deux façons de noter les attributs?

Created by Swen-Peter Ekkebus
from the Noun Project

CHP15 DIAGRAMME DE CLASSE

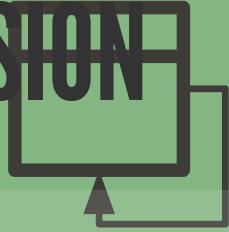


1. Quelle est la différence entre un diagramme de classes conceptuelles et logicielles?
2. Quelles sont les deux façons de noter les attributs?
3. Pourquoi fait-on un DCC?

Created by Swen-Peter Ekkebus
from the Noun Project

14 . 17

VALIDATION DE LA COMPRÉHENSION



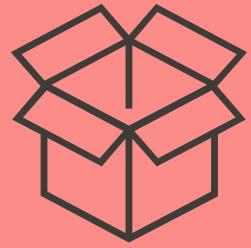
- Combien d'attributs pour Register et Sale ?
 - A. 1, 1
 - B. 2, 1
 - C. 1, 2
 - D. 2, 2

ARCHITECTURE EN COUCHE

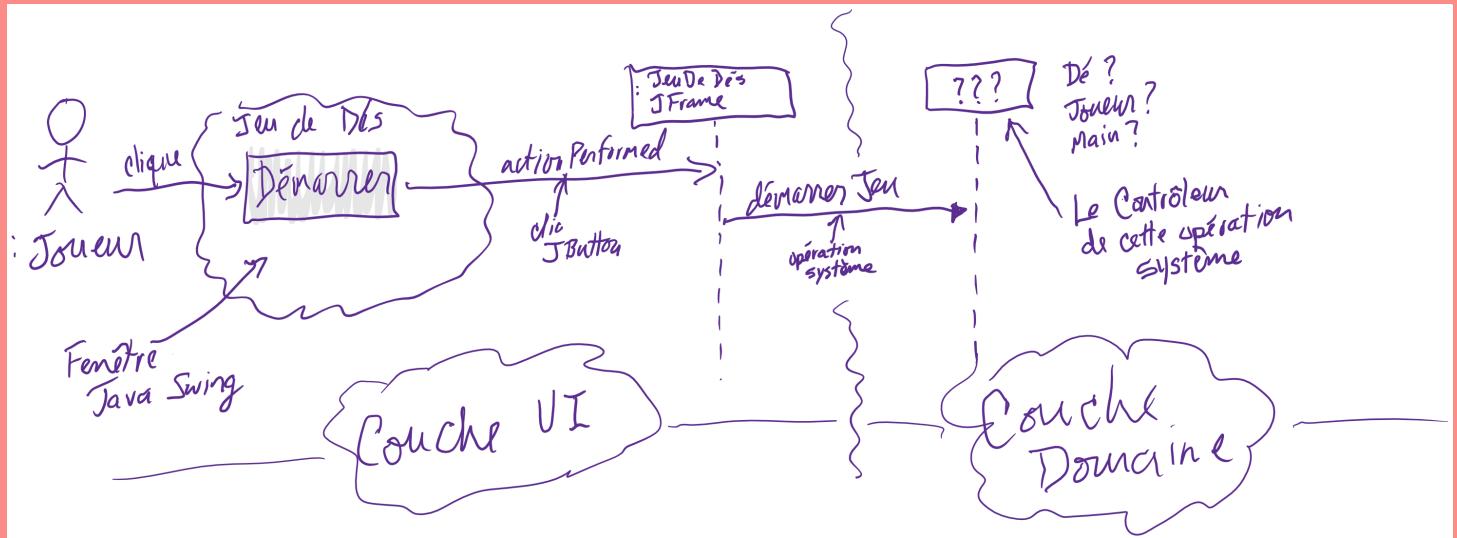
1. Architecture logique en couches - 21.45m
2. Packaging - 34.58m
3. Valider l'architecture en couche du laboratoire - 10.19m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

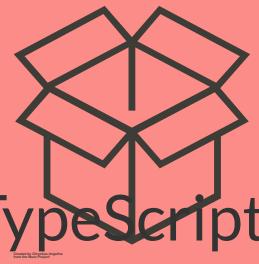
ARCHITECTURE LOGIQUE



Solution avec Java/Swing:



ARCHITECTURE LOGIQUE



Solution avec Navigateur/HTTP/ExpressJS/TypeScript:

?

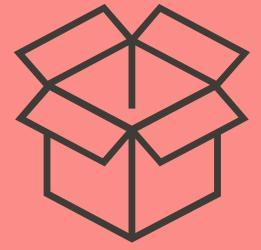
15 . 3

COMPRENDRE ET RESPECTER L'ARCHITECTURE EN COUCHES DU LABORATOIRE



<https://log210-cfuhrman.github.io/log210-valider-architecture-couches/>

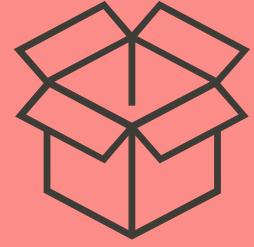
PACKAGING?



15 . 5

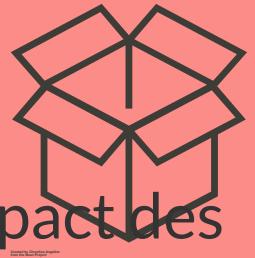
PACKAGING?

ATTRIBUT DE QUALITÉ



- Performance
- Disponibilité
- Testabilité
- Interopérabilité
- Sécurité
- Usabilité
- Modifiabilité

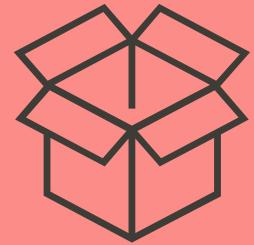
OBJECTIFS



- Organiser les packages pour réduire l'impact des modifications
- Apprendre d'autres notations des structures de packages UML

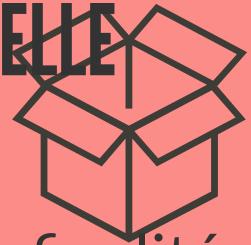
15 . 6

PRINCIPES



- Organiser les packages par **cohésion**
- Packager une **famille d'interface**
- Créer un package par **tâche** et par **groupe de classes instables** (Branches)
- Les plus responsables sont les plus stables
- Factoriser les type indépendants
- Utiliser des **fabrications** pour limiter la dépendance aux packages concrets
- Pas de cycles dans les packages **DSM**

GROUPEMENT PAR COHÉSION FONCTIONNELLE

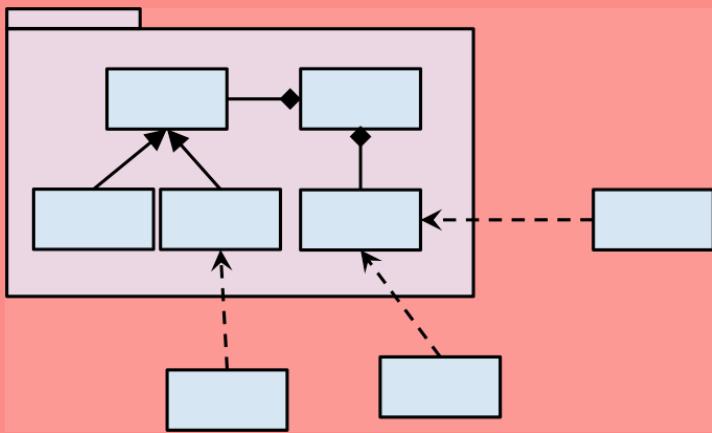


- On groupe les types qui sont fortement apparentés, parce qu'ils participent à une finalité, un service, des collaborations, une politique et une fonction qu'ils ont en commun.
- P.ex. le package "Tarification" contient des classes liées à la tarification des produits.
- C'est intuitif.

GROUPEMENT PAR COHÉSION RELATIONNELLE

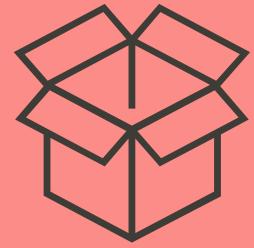


- Mettre dans un package des classes qui se sont fortement couplées les unes aux autres.
- $CR = (\text{NombreDeRelationsInternes} + 1) / \text{NombreDeTypes}$



- GOF Facade pour améliorer la cohésion relationnelle avec les packages externes

COHÉSION RELATIONNELLE - EXEMPLE



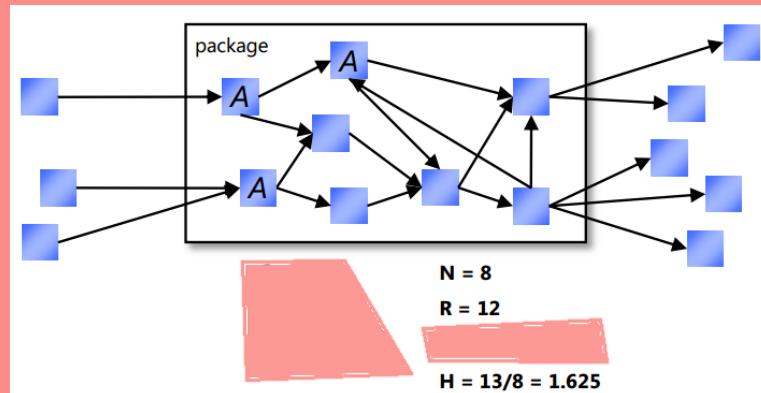
cohesion

Relational Cohesion (H): average number of internal relationships per type:

$$H = (R + 1) / N, \text{ where}$$

R = number of type relationships internal to the package,
and

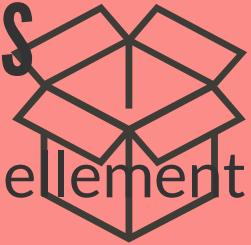
N = number of types in the package.



ndepend: métrique de cohésion

15 . 10

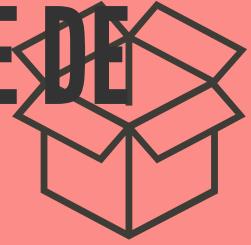
PACKAGER UNE FAMILLE D'INTERFACES



- Placez une famille d'interfaces fonctionnellement liées dans un package à part
- Cela vaut pour les familles de trois interfaces ou plus
- Exemple: javax.ejb comprend au moins douze

```
- EJBContext
- EJBHome
- EJBLocalHome
- EJBLocalObject
- EJBMetaData
- EJBObject
- EnterpriseBean
- EntityBean
- EntityContext
- Handle
- HomeHandle
- MessageDrivenBean
- MessageDrivenContext
- SessionBean
- SessionContext
- SessionSynchronization
- TimedObject
- Timer
- TimerHandle
- TimerService
```

CRÉEZ UN PACKAGE PAR TÂCHE DE LIVRAISON



- Les packages sont l'unité de base de développement et de livraison (on ne développe rarement qu'une seule classe)
- Créer un package par tâche de livraison



15 . 12

... ET PAR GROUPE DE CLASSES INSTABLES [2/3]



- Si un ensemble de classes dans un package a une tendance à évoluer ensemble, p.ex. on remarque qu'à chaque itération, c'est toujours les mêmes 5 ou 10 classes dans un package qui ont été modifiées.
- Exemple concret – chaque fois que l'on change le schéma de la BD il y a des changements dans plusieurs classes, surtout les appels à la méthode “sqlquery()”
 - Créez un package dans le package (sous package) pour isoler les classes ayant une tendance à changer ensemble
 - Notez que la tendance de ce genre de changement n'est pas visible dans les premières itérations.

LES PLUS RESPONSABLES SONT LES PLUS STABLES

Les packages « les plus responsables » engendrent le plus de dépendances et devraient être plus stables.

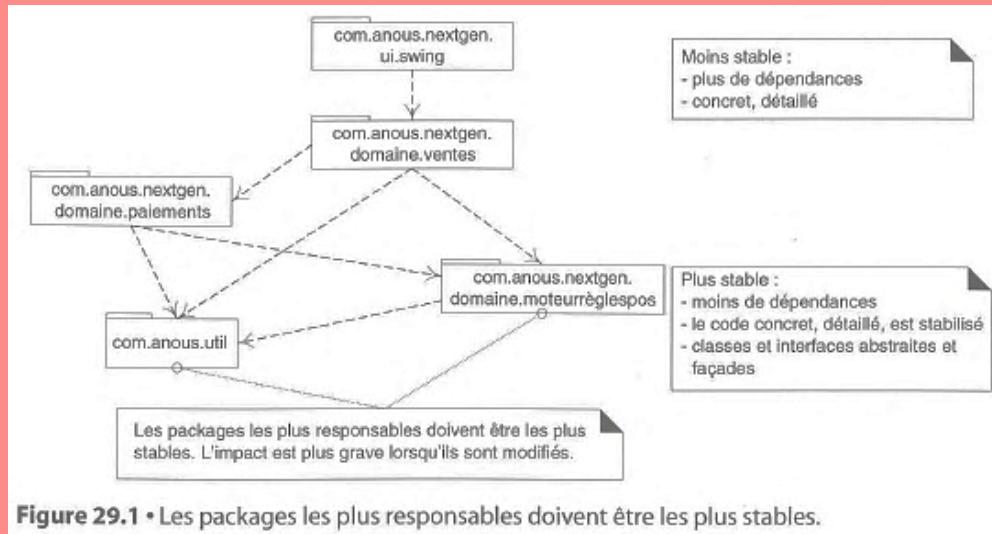


Figure 29.1 • Les packages les plus responsables doivent être les plus stables.

LES PLUS RESPONSABLES SONT LES PLUS STABLES

Les packages « les plus responsables » engendrent le plus de dépendances et devraient être plus stables.

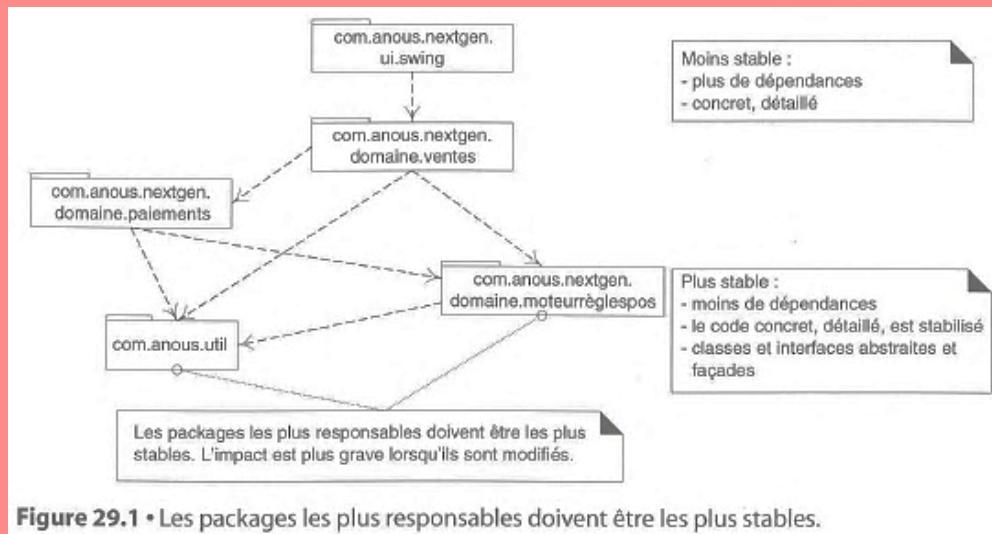


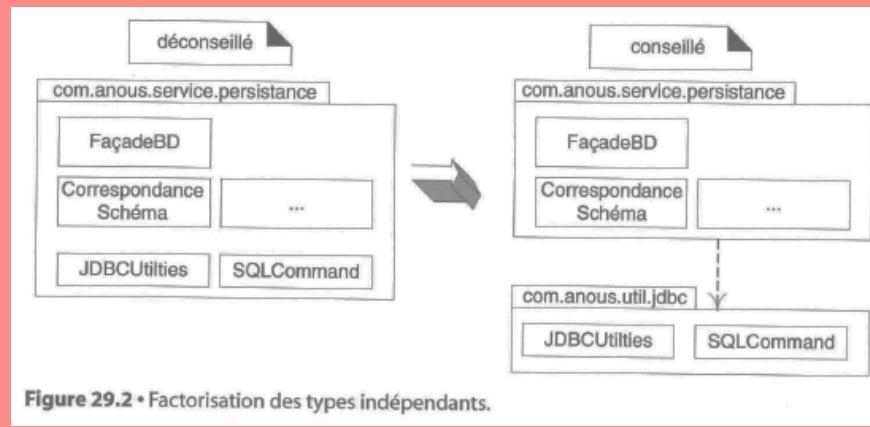
Figure 29.1 • Les packages les plus responsables doivent être les plus stables.

Quel GRASP pour rendre un package plus stable?

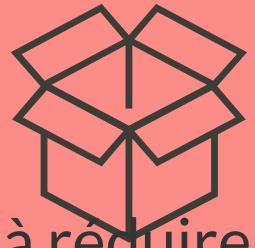
FACTORIZER LES TYPES



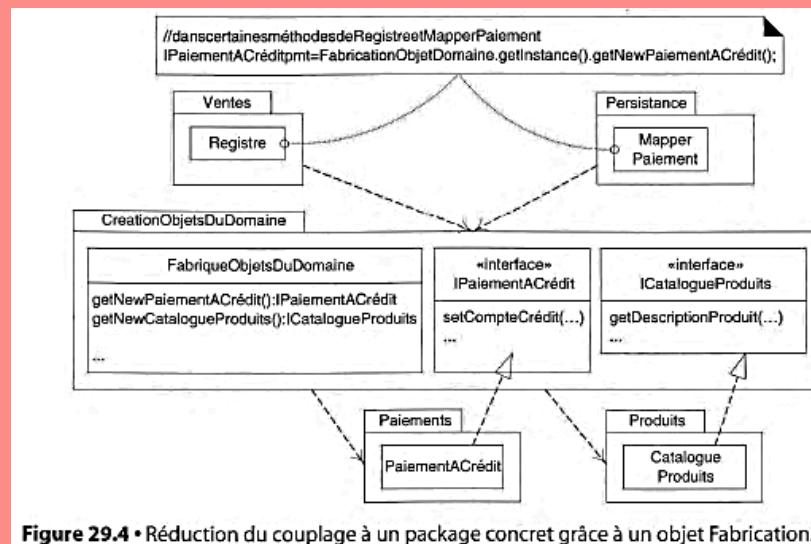
Organiser les types qui peuvent être utilisés indépendamment ou dans des contextes différents en packages distincts.



FABRIQUES POUR DÉCOUPLER DES CLASSES CONCRÈTES

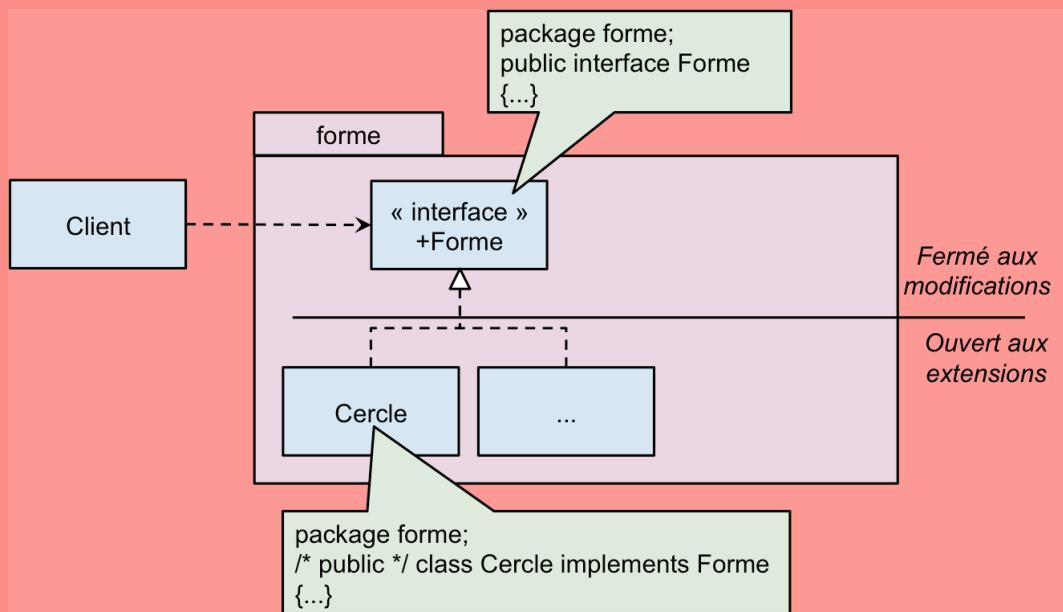


Renforcer la stabilité des packages consiste à réduire leur dépendance à des classes concrètes qui appartiennent à d'autres packages



VISIBILITÉ ET ENCAPSULATION DANS LES PACKAGES

Organiser les classes dans un package avec la bonne visibilité



VISIBILITÉ ET ENCAPSULATION DANS LES PACKAGES



- Bloquer l'accès à un attribut ou une méthode d'une classe avec `private`
- Bloquer l'accès à une classe dans un package?
 - Déclarer la classe `class X { ... }` (package protection, sans mot clé)
 - GRASP Protection des variations
 - Client ne devrait pas accéder aux implémentations de Forme.

VISIBILITÉ ET ENCAPSULATION DANS LES PACKAGES



- Cette visibilité empêche les transgressions de l'encapsulation

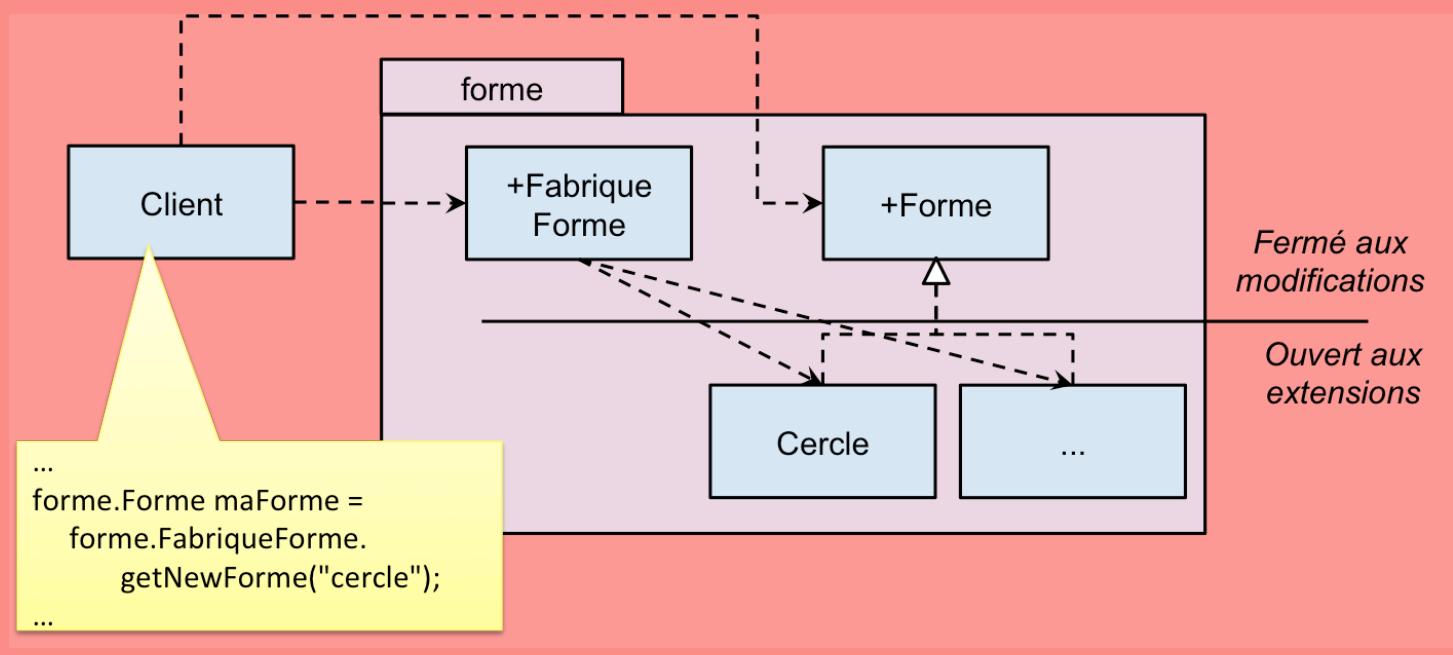
```
package client;

public class Client {
    /**
     * @param args
     */
    public static void main(String[] args) {
        forme.Forme maForme = new forme.Cercle();
        maForme.dessiner();
    }
}
```

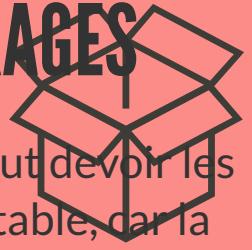
COMMENT INSTANCIER LES CLASSES INVISIBLES?



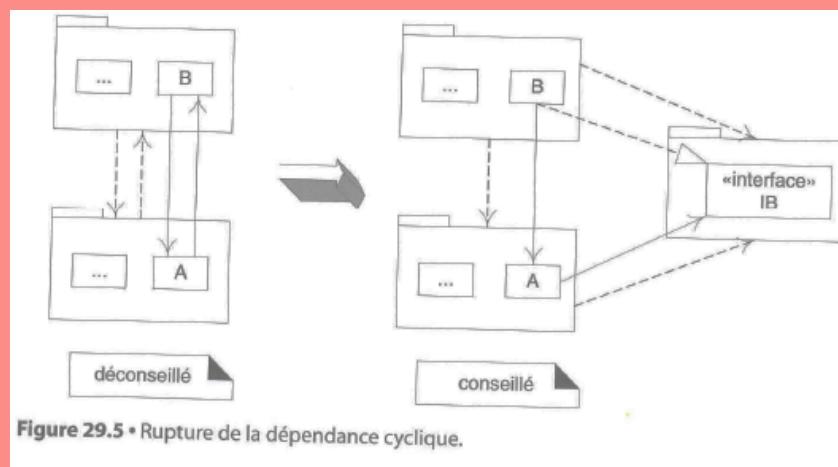
- FabriqueForme permet à Client d'instancier les Formes concrètes, bien qu'elles lui soient invisibles



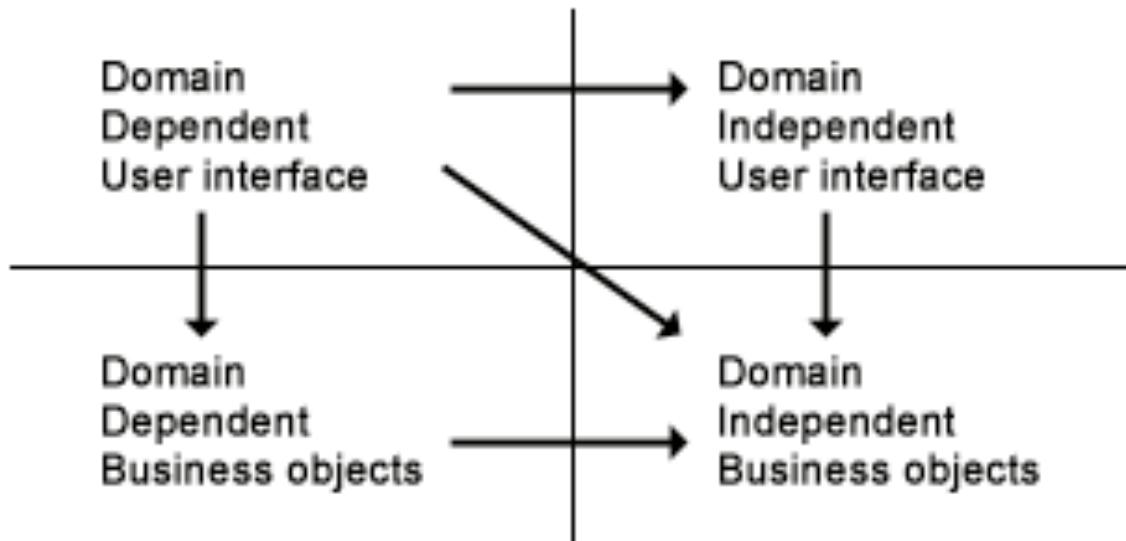
PRINCIPE: PAS DE CYCLES DANS LES PACKAGES



- Si un groupe de packages a une dépendance cyclique, on peut le traiter comme un seul grand package. Cela n'est pas souhaitable, car la livraison de packages volumineux (ou d'agrégats de packages) aggrave le risque d'effets néfastes.
- Factoriser les types participant au cycle dans un package plus petit.
- Rompre le cycle à l'aide d'une interface.



COMBINAISON DES PATRONS DE PACKAGING



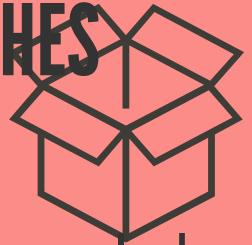
ref: Commonly used architectural patterns in Java applications

QU'EST-CE QU'UNE COUCHE?



- Groupement à forte granularité de classes, packages et sous-systèmes
- Possède un thème « cohésif »
- Une couche plus « haute » fait appel aux couches plus « basses », mais pas l'inverse
- Couches OO typiques
 - Présentation (IHM)
 - Logique applicative (objets du domaine)
 - Services techniques (persistance, journalisation, etc.)

CONCEVOIR UNE ARCHITECTURE EN COUCHES



- Séparation nette et cohésive
- Collaboration et couplage vont du haut vers le bas

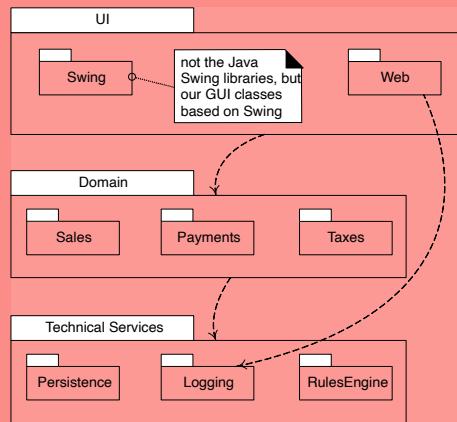


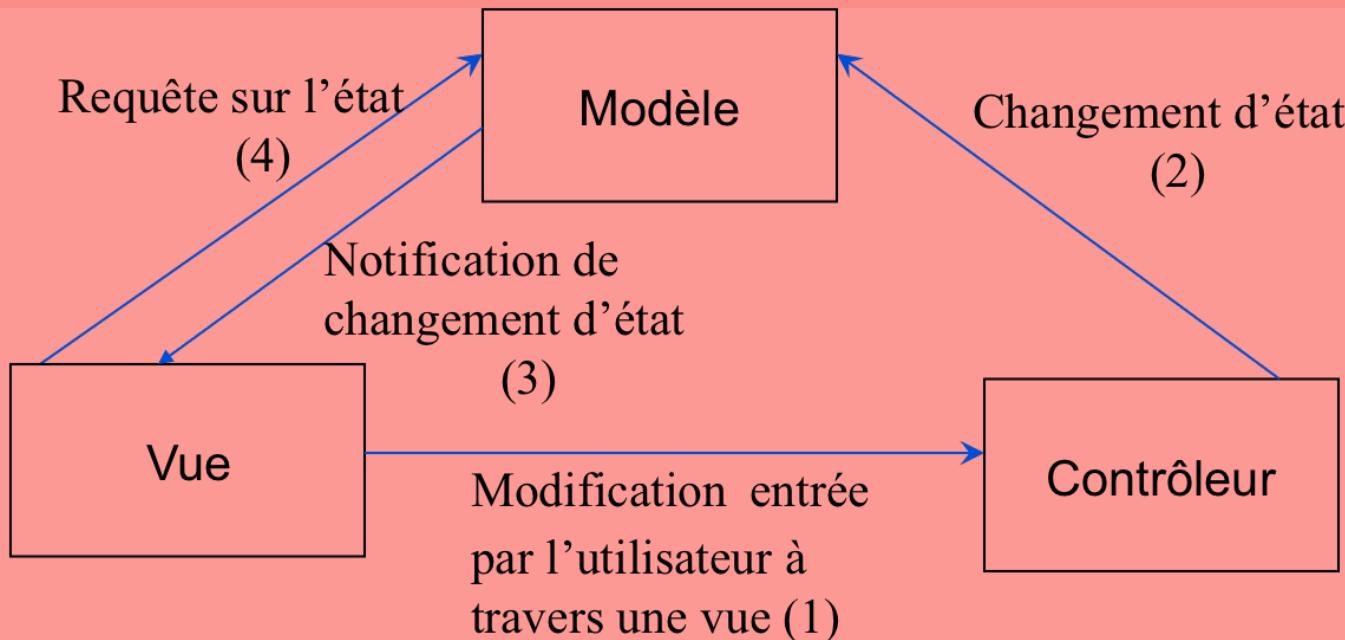
fig F12.2, A13.2

- Séparation Modèle-Vue est un cas simple

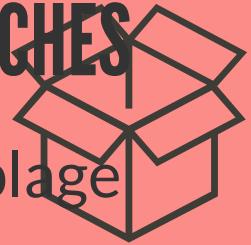
ARCHITECTURE - MODÈLE / VUE / CONTRÔLEUR



- Vues/contrôleurs modifient le modèle
- Modèle informe les vues des changements
- Vues se mettent à jour de nouveau en conséquence

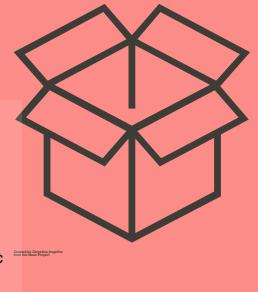
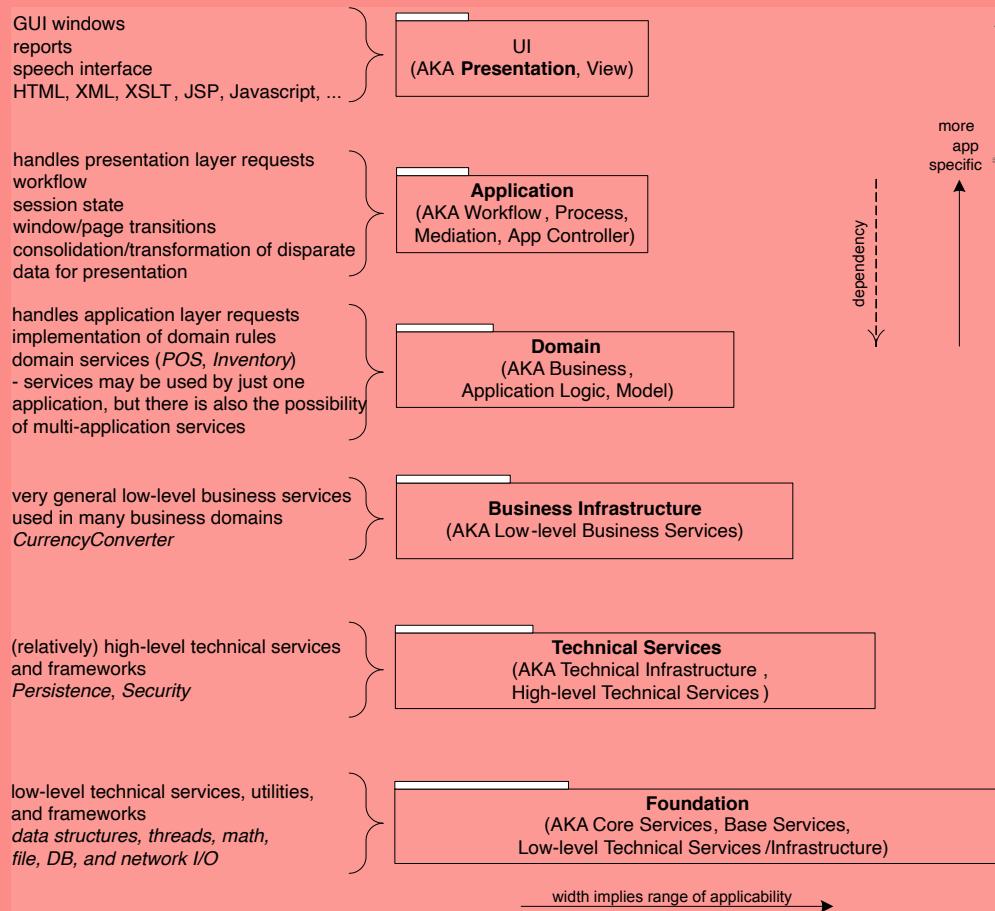


AVANTAGES DE LA MODÉLISATION EN COUCHES



- Séparation de fonctions/services -> couplage réduit, cohésion améliorée
- La complexité est encapsulée (décomposable)
- On peut remplacer certaines couches
- Les couches basses contiennent des fonctions réutilisables
- Certaines couches (Domaine) peuvent être distribuées
- Segmentation logique facilite développement en équipe

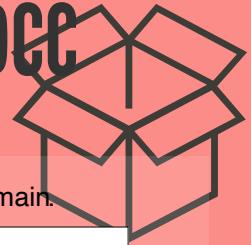
ARCHITECTURE MULTI-COUCHES



more app specific

dependency

RELATION ENTRE COUCHE DOMAINE ET DCC

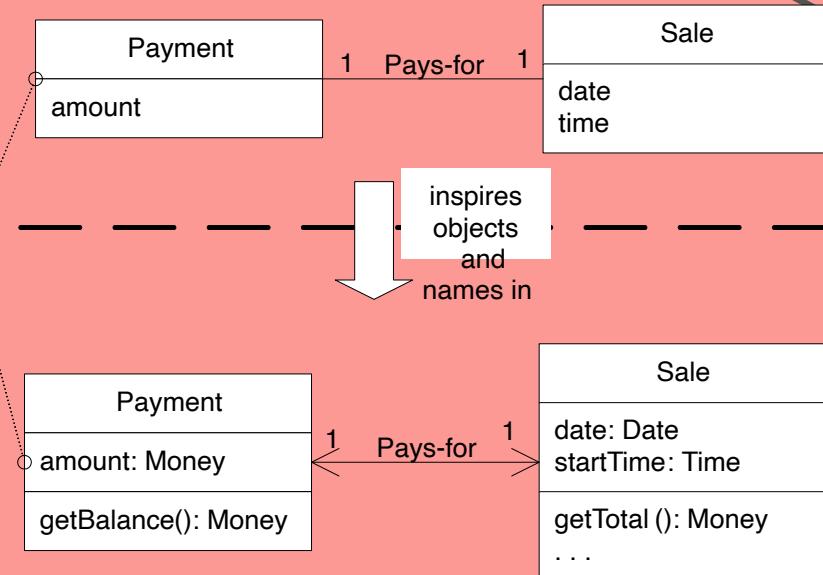


UP Domain Model
Stakeholder's view of the noteworthy concepts in the domain.

A Payment in the Domain Model is a concept, but a Payment in the Design Model is a software class. They are not the same thing, but the former *inspired* the naming and definition of the latter.

This reduces the representational gap.

This is one of the big ideas in object technology.

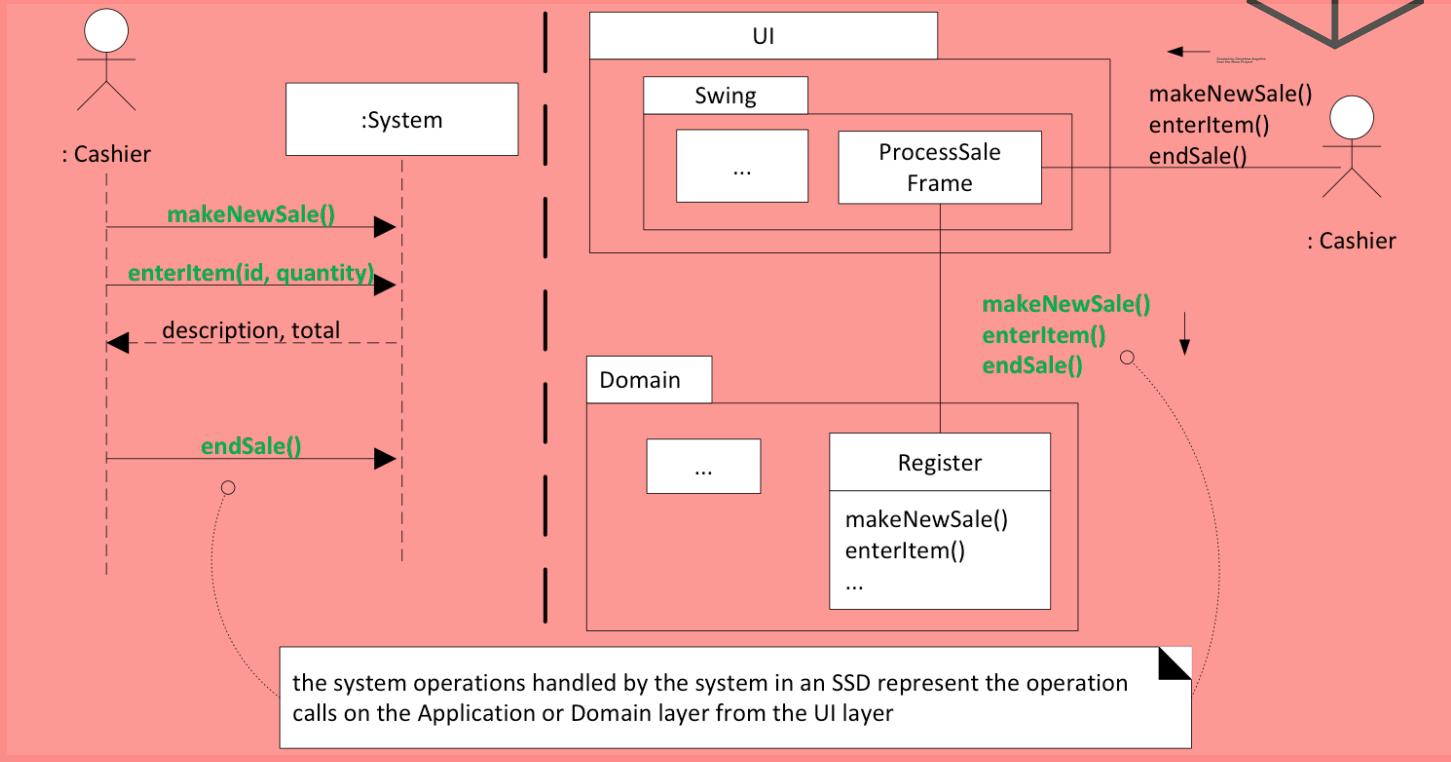
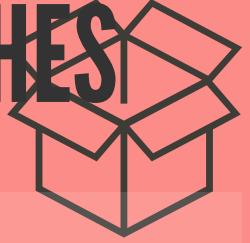


Domain layer of the architecture in the UP Design Model

The object-oriented developer has taken inspiration from the real world domain in creating software classes.

Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

RELATION ENTRE DSS ET COUCHES



INFLUENCE DES ARTEFACTS DU PROCESSUS UNIFIÉ

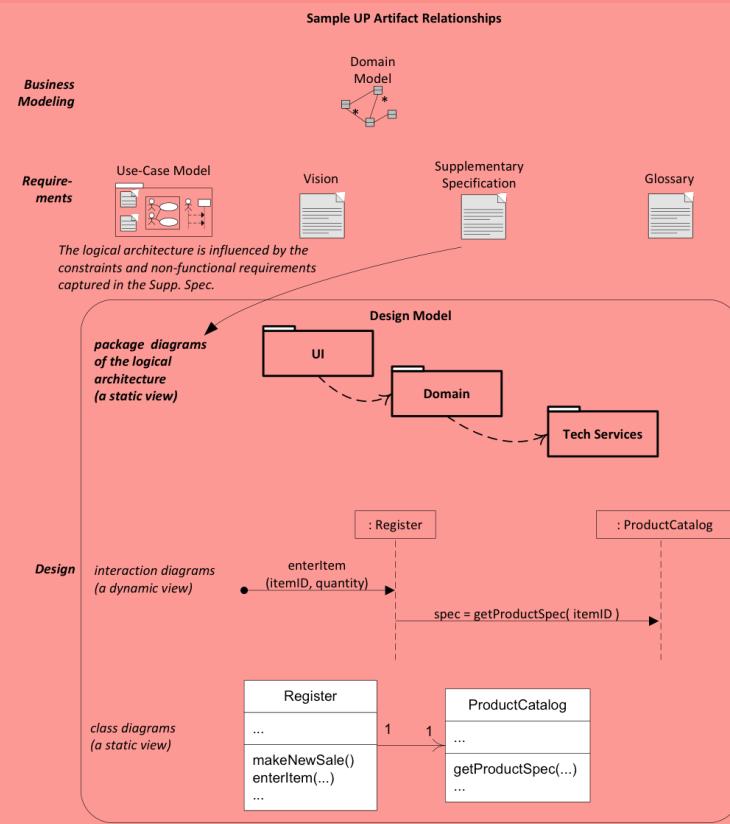
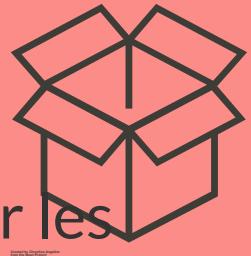


fig A13.1, F12.1

RÉSUMÉ



- Architecture logique est importante pour les conceptions de systèmes complexes
- Modèle en couches est populaire
 - Extension du principe de séparation Modèle-Vue
 - Java est organisé en couches
 - Android aussi
- DSS relie la couche présentation avec la couche du domaine à travers les opérations système

VALIDER L'ARCHITECTURE EN COUCHE



Valider l'architecture en couches (des laboratoires)

15 . 32

MODULE EXERCICES

1. Git exercices
2. Mise en plateau - Google Doc
3. Reserver plusieurs chambres
4. Système d'échange de livre universitaire
5. Exercice Système d'échange de livre - 55.43m
6. Exercices de préparation à l'examen final
7. Patron observateur

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

EXERCICE MISE EN PLATEAU </>

[https://docs.google.com/document/d/1nGiQOeRyrY-Tx-J82z3qa0vAoUsF1rvHMNjyPGXH2mY/edit?
usp=sharing](https://docs.google.com/document/d/1nGiQOeRyrY-Tx-J82z3qa0vAoUsF1rvHMNjyPGXH2mY/edit?usp=sharing)

16 . 2

ITÉRATION SUIVANTE

CU02-Le client aimerais pouvoir réserver plusieurs chambres avec des dates différents lors de la même réservation

- Qu'est-ce qui change au niveau du DCU
- Qu'est-ce qui change au niveau du CU
- Qu'est-ce qui change au niveau du MDD
- Qu'est-ce qui change au niveau des IU
- Qu'est-ce qui change au niveau du Dss
- Qu'est-ce qui change au niveau des contrats
- Qu'est-ce qui change au niveau des RDCU
- Qu'est-ce qui change au niveau du DCL

Noter une réservation

--

EXERCICE

SYSTÈME ÉCHANGE DE LIVRE

- En équipe de 4 personnes, réaliser les modèles suivant:
 - MDD(avec catégorie de classe et d'association),
 - DSS,
 - Contrats,
 - RDCU
 - DCL

16 . 3

ÉTUDE DE CAS



SYSTÈME D'ÉCHANGE DE LIVRES UNIVERSITAIRES

Le Bureau du Développement Durable (BDD) de l'Université a mis en place un système d'échange de livres aux fins de développement durable et pour réduire les coûts pour les étudiants (les clients du système). La version initiale est rudimentaire et ne permet que deux fonctionnalités :

CU01 - AJOUTER UN LIVRE À ÉCHANGER



Acteur principal : Client (étudiant)

Préconditions :

Le Client est identifié (par son nom d'utilisateur) et authentifié par son mot de passe.

Scénario principal (succès)

1. Le Client démarre un nouvel ajout de livre.
2. Le Client entre le code ISBN du livre, ainsi que le code de sa condition.
3. Le Système enregistre le livre et présente sa description (titre, nombre de pages, auteurs, maison d'édition, no d'édition). Le Client répète les étapes 2 à 3 jusqu'à ce qu'il ait saisi tous les livres à échanger.
4. Le Système présente la liste de livres que possède le Client.

Cas alternatifs

- 3a. Le système affiche un message d'erreur puisque le livre n'existe pas.

CU02 - PROPOSER UN ÉCHANGE DE LIVRES



Acteur principal : Client (étudiant)

Préconditions : Le Client est identifié (par son nom d'utilisateur) et authentifié par son mot de passe.

Scénario principal (succès)

1. Le Client démarre une nouvelle proposition d'échange de livres.
2. Le Système présente une liste d'autres Clients dans le système ainsi que le(s) livre(s) qu'ils ont à échanger.
3. Le Client sélectionne un autre Client (le Client Proposé) à qui il veut proposer un échange.
4. Le Système présente la liste de livres que possède le Client Proposé et une liste de livres que possède le Client.
5. Le Client ajoute à la proposition d'échange un livre. Si c'est un de ses livres, alors c'est à offrir dans la proposition. Sinon c'est un livre du Client Proposé et c'est à recevoir dans la proposition. Le Client répète l'étape 5. jusqu'à ce qu'il soit satisfait de la proposition.
6. Le Système présente le nombre total de livres à offrir et à recevoir et demande au Client de confirmer la proposition.
7. Le Client confirme et le Système enregistre sa proposition d'échange avec la date et l'heure.
8. Le Système envoie un courriel au Client Proposé pour l'informer de la proposition d'échange.

RÉVISION EXERCICE

SYSTÈME D'ÉCHANGE DE LIVRE

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

EXERCICES EN LIGNE

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

16 . 8

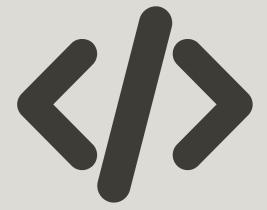
MODULE TYPESCRIPT

1. Exemples TypeScript (SGB) - 24.55m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

17 . 1

EXEMPLES TYPESCRIPT



Dépôt github

<https://github.com/profcfuhrmanets/exemples-ts/>

17 . 2

ARROW FUNCTION (JAVASCRIPT) </>

src/arrow_function.ts

17 . 3

ARRAY.PROTOTYPE.MAP() (JAVASCRIPT)

src/array_map_arrow.ts

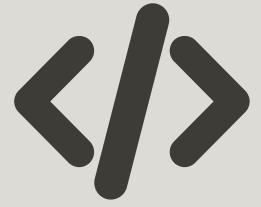
17 . 4

TABLEAU ASSOCIATIF MAP() (JAVASCRIPT)

src/tableau_associatif.ts

17 . 5

ACCÈS SGB



src/get_students_fetch.ts

17 . 6



DEUX “MAP”



Le sens est différent:

- Tableau associatif Map<>()
 - <https://howtodoinjava.com/typescript/maps/>
- Array.prototype.map()
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

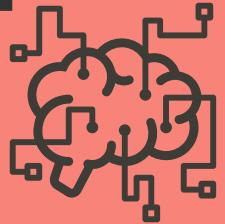
MODULE COMPLEXITÉ

1. Complexité - Inhérente et accidentelle - 23.04m

Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DCL | Couche | Exercice | Typescript | Complexite | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité | Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

18 . 1

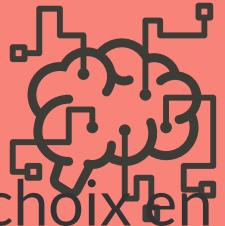
COMPLEXITÉS INHÉRENTE ET ACCIDENTELLE



Created by Weltenraser
from the Noun Project

18 . 2

OBJECTIF



Vous sensibiliser aux conséquences de vos choix
en conception sur le plan de la complexité.

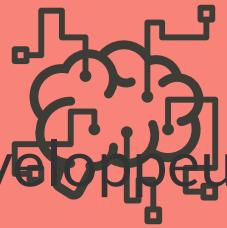
Created by Weltenrazer
from the Noun Project

**Principe
KISS**



Décorateur,
Observateur, MVC,
Héritage,
Polymorphisme,
ArrayList<MaClasse>

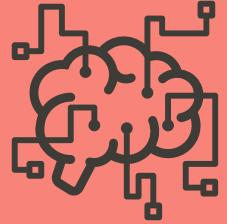
DÉFIS POSÉS PAR LA COMPLEXITÉ



Created by Weltenraiser
from the Noun Project

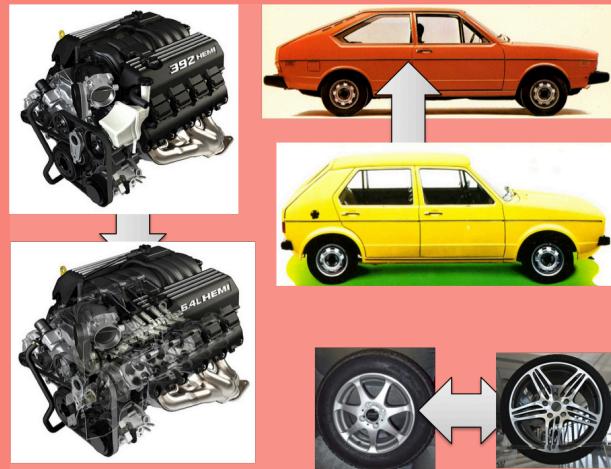
- La complexité est l'adversaire de tout développeur logiciel
- On peut distinguer entre deux sources de complexité (F. Brooks)
- Le domaine du problème
 - « complexité inhérente (essentielle) »
- La solution (logicielle)
 - « complexité accidentelle »

COMPLEXITÉS



Created by Weltenrazer
from the Noun Project

- Inhérente (essentielle)
 - « Essentielle » selon Aristote
 - Pour « être » une voiture, une voiture a un moteur, des roues, des portes, etc.
- Accidentelle (due à des choix)



18 . 5

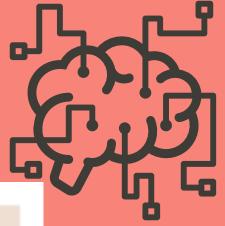
COMPLEXITÉ INHÉRENTE



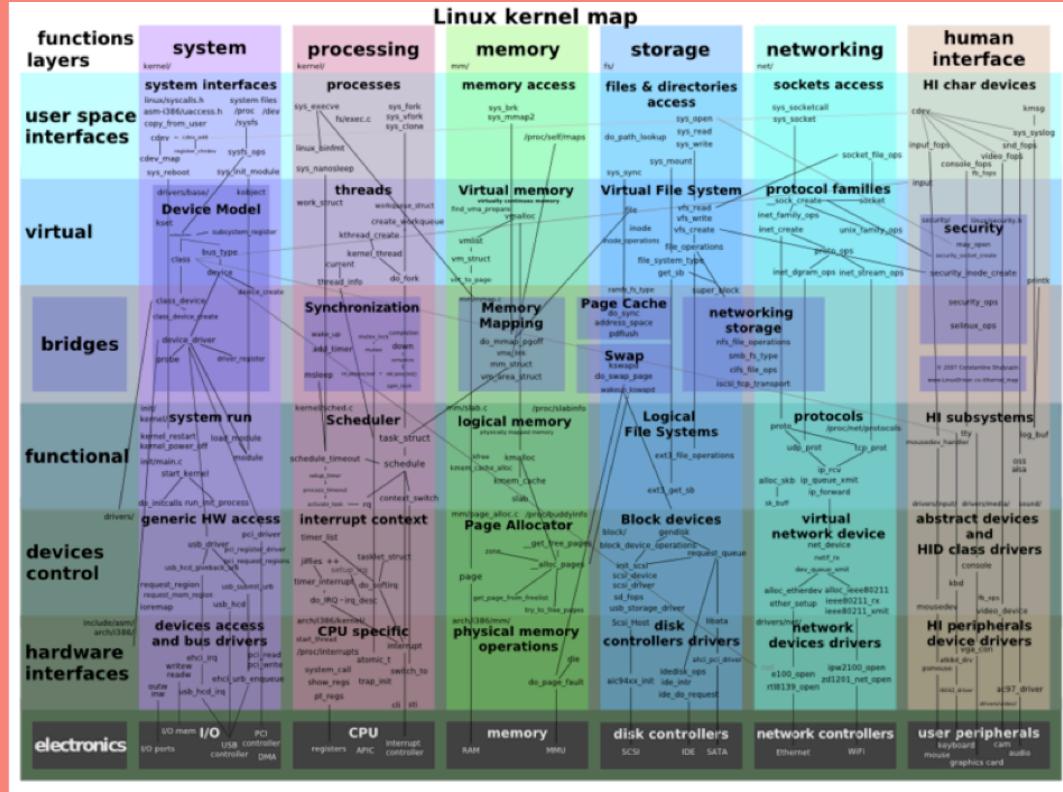
Created by Weltenrazer
from the Noun Project

- Le domaine du problème concernant le logiciel peut être complexe
 - Exemple : système d'exploitation pour ordinateur
- La complexité inhérente est due au « problème » et non à la solution
 - Linux est complexe parce qu'il doit supporter plein de fonctionnalités...
- Synonyme : « complexité essentielle »

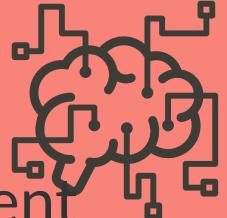
COMPLEXITÉ INHÉRENTE



by Weltenraser
The Noun Project



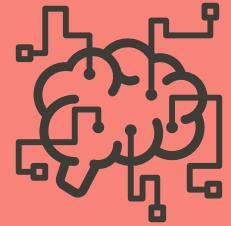
COMPLEXITÉ ACCIDENTELLE



Created by Weltenraiser
from the Noun Project

- Les choix proposés pour la solution peuvent introduire une complexité « accidentelle »
 - Langage de développement à bas niveau (p. ex. C ou assembleur)
 - Outils de débogage limités
- La complexité accidentelle peut être plus facilement gérée
 - Meilleurs outils (IDE), meilleures techniques, langage plus haut niveau, solution plus simple (si possible)

EXEMPLE : GESTION DE COMPLEXITÉ ACCIDENTELLE EN JAVA



- Compilation en ligne de commande
- Utilisation de IDE (Eclipse)

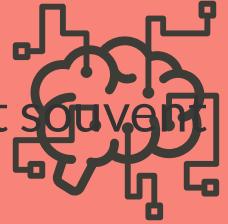
```
C:\>javac PingPong.java
PingPong.java:104: error: unreported exception
InterruptedException; must be caught or declared to
be thrown
                                lock.wait();
                                         ^
1 error
```

A screenshot of an IDE showing Java code. A tooltip is displayed over the line `lock.wait();`. The tooltip contains two options: "Add throws declaration" and "Surround with try/catch". The code snippet is as follows:

```
for (int i = 0; i < LOOP_COUNT; i++) {
    synchronized(lock) {
        while (!lock.waitingFor==PONG) {
            lock.wait(); // wait for Ping to print
        }
    }
}
```

...
// Pong thread logic
private void doPong() throws InterruptedException {
 ...

DISTINCTION N'EST PAS TOUJOURS FACILE



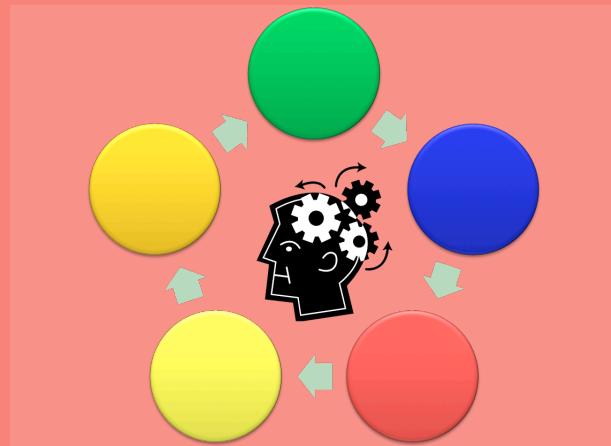
- Dans une conception logicielle, les éléments sont souvent complexes.
- La source de la complexité n'est pas toujours facile à identifier.
 - Exemple: logiciels de réseau
 - Difficultés rencontrées: délais, pertes d'informations, timeout, communication asynchrone , etc.
 - Sont-elles dues aux complexités inhérentes ou accidentielles?
- Communication asynchrone vs complexité inhérente
- Pertes d'informations vs complexité accidentelle (p.ex. avec protocole UDP)

GÉRER LA COMPLEXITÉ



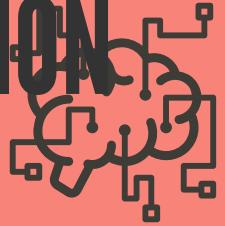
Created by Weltenraiser
from the Noun Project

- Réduire l'effort cognitif en gérant la complexité inhérente
 - Abstractions (OO)
 - Encapsulation (OO)
 - Masquage de l'information (OO)



18 . 11

ABSTRACTION ET ENCAPSULATION

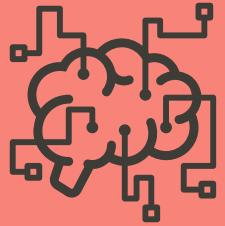


Created by Weltenraiser
from the Noun Project

18 . 12

ABSTRACTION

Vue plus simple



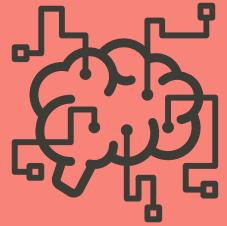
Created by Weltenraiser
from the Noun Project



18 . 13

ENCAPSULATION

Accès limité



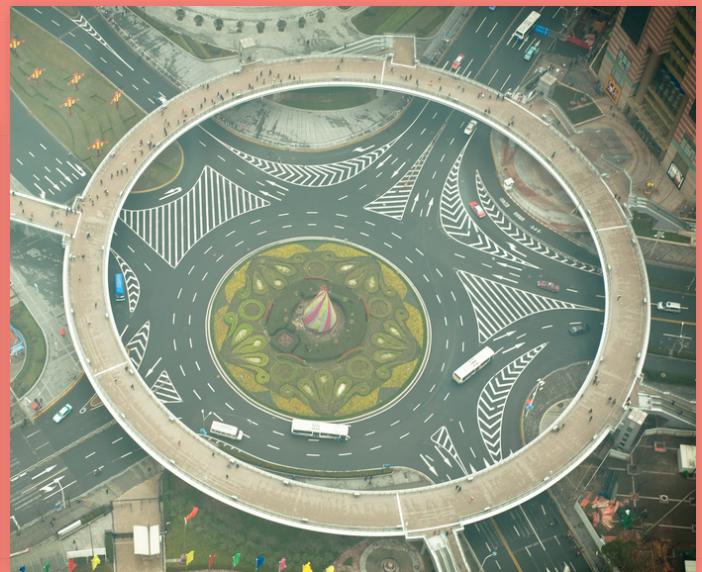
Created by Weltenraiser
from the Noun Project



PATTERNS POUR GÉRER LA COMPLEXITÉ

- Proposer des solutions aux problèmes récurrents dans un contexte particulier

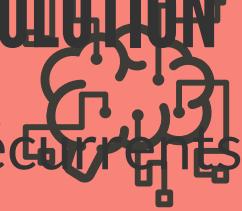
Created by Weltenraiser
from the Noun Project



DESIGN PATTERNS SONT UNE PARTIE DE LA SOLUTION

- Proposer des solutions aux problèmes rencontrés dans un contexte particulier

Created by Weltenraiser
from the Noun Project



Appareils mobiles



Avionique



Automobile

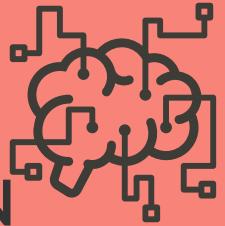


SIGNETS

Guichet interactif

SOCRATIVE OU ZOOM

Nom de la salle de class: ETSDESIGN



Created by Weltenraser
from the Noun Project

A screenshot of a web browser window showing the Socrative student login interface. The URL in the address bar is <https://b.socrative.com/login/student/>. The page features a light blue header with the Socrative logo and the text "Connexion élève". Below this is a form with a text input field labeled "Nom de la salle de classe" and an orange "REJOINDRE" button. At the bottom right of the form is a language selection dropdown set to "Français (Canadien)". The browser's title bar shows "Socrative".

18 . 17

VALIDATION DE LA COMPRÉHENSION



Created by Weltenraiser
from the Noun Project

- En développement logiciel, la complexité inhérente est due au domaine du problème et cause des complexités dans la solution.
 - Vrai?
 - Faux?

VALIDATION DE LA COMPRÉHENSION



Created by Weltenraiser
from the Open Project

- La complexité inhérente est de mieux en mieux gérée grâce aux innovations technologiques.
 - Vrai?
 - Faux?

VALIDATION DE LA COMPRÉHENSION



Created by Weltenraser
from the Noun Project

- Le déverminage d'un algorithme de parcours d'arbre représentant une expression mathématique est difficile à cause de la complexité accidentelle
 - Vrai?
 - Faux?

VALIDATION DE LA COMPRÉHENSION



Created by Weltenraiser
from the Noun Project

- Les outils de développement comme Eclipse répondent principalement aux difficultés dues à la complexité inhérente.
 - Vrai?
 - Faux?

18 . 21

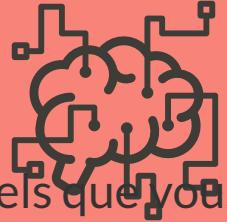
VALIDATION DE LA COMPRÉHENSION



Created by Weltenraiser
from the Noun Project

- Qu'est-ce qui n'est pas une technique pour gérer la complexité inhérente?
 1. Masquage de l'information
 2. Encapsulation
 3. Ramasse-miettes (récupérateur de mémoire)
 4. Abstraction
 5. Généralisation

RÉSUMÉ



Created by Weltenraiser
from the Noun Project

- Soyez attentif aux sources de complexité dans les logiciels que vous développez
- La complexité inhérente ne peut pas être éliminée si votre problème est complexe
- Cependant, elle peut être gérée avec un bon choix d'abstractions.
- Patrons utilisent des abstractions et de l'encapsulation pour gérer la complexité
- Cependant, ils amènent aussi de la complexité accidentelle
- Vérifier:
 - Le problème récurrent que le patron est censé résoudre existe-t-il? (facile)
 - Le coût de la complexité due au patron est acceptable par rapport aux bénéfices (moins facile)

MODULE EXIGENCES FURPS

1. FURPS - 39.44m

Séances | Quiz | Intro | Outils | ACOO | Équipe | PU | DCU | CU | IU | MDD | Dss | Contrat | RDCU | DCL | Couche | Exercice | Typescript | Complexité | FURPS | TDD | GOF | Diagramme d'état | Diagramme d'activité | Diagramme composant et déploiement | Diagrammes | Dette technique | Divers

19 . 1

FURPS - EXIGENCES DU CLIENT



Created by Bharat
from the Noun Project

- FURPS+

1. Fonctionnalité (Use case)
2. Usability (Aptitude à l'utilisation, facteurs humains, aide et documentation)
3. Reliability (Fiabilité)
4. Performance
5. Supportability (Possibilités de prise en charge, adaptabilité, maintenabilité, etc.)

FURPS ET PATRONS



Created by Bharat
from the Noun Project

L'application d'un patron doit être motivée par des exigences:

- *Facade* favorise la modifiabilité du code (**S**)
- *Memento* favorise l'implémentation de « Undo » (**F** et **U**)
- *Stratégie* favorise des « plug-in » (**S**)
- *Stratégie* peut faciliter la mise en place de la redondance de l'information (**R** et **P**)
- *Flyweight* favorise une meilleure performance (**P**)

DÉFINITION DES BESOINS

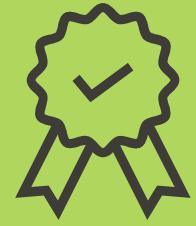


- Cas d'utilisation
 - Descriptions des besoins fonctionnels
 - Le « F » dans FURPS

Created by Bharat
from the Noun Project

ÉNONCÉ LABORATOIRE

Le modèle FURPS est utilisé.



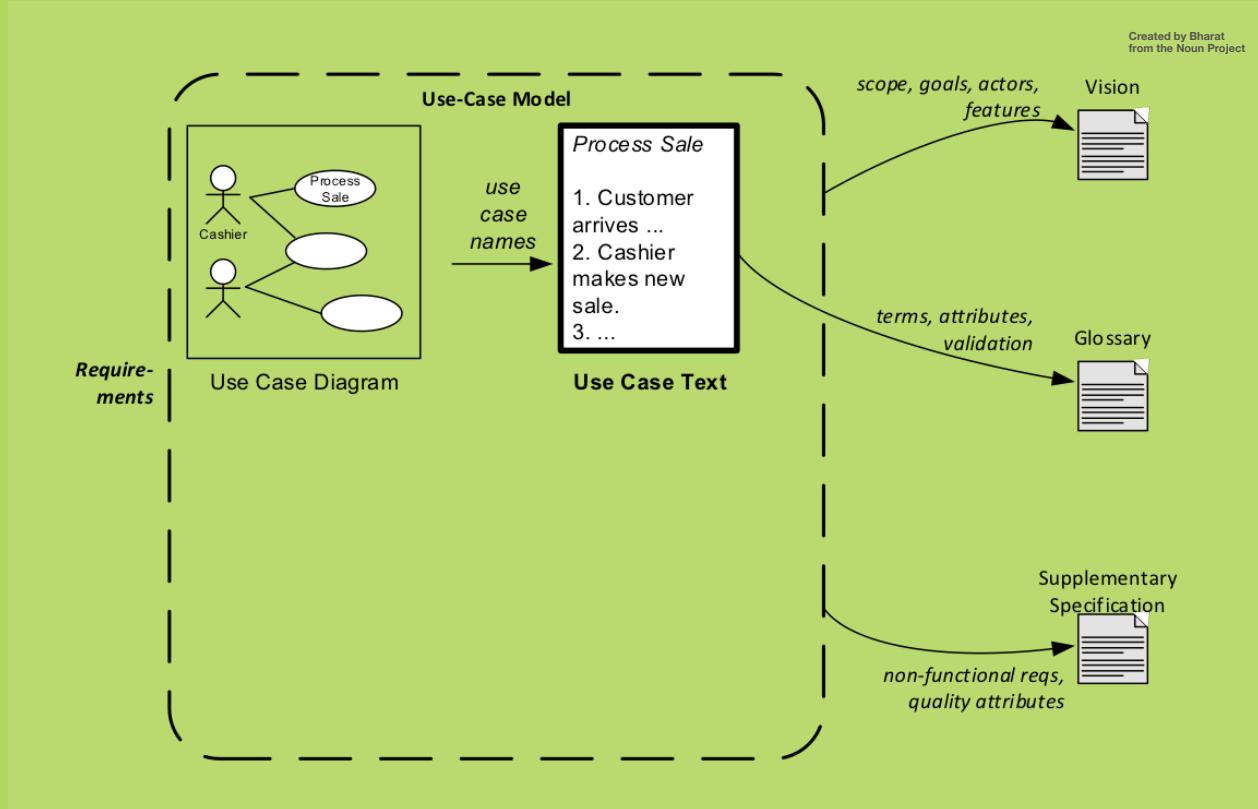
Created by Bharat
from the Noun Project

Plus d'exemples.

19 . 5

COMMENT ORGANISER LES BESOINS DANS LE PU?

- Plusieurs artefacts décrivent les besoins



EXIGENCES FURPS+



Created by Bharat
from the Noun Project

- Une messagerie en nuage
- Calendriers et contacts partagés
- Messagerie instantanée, appels de PC à PC et conférence en ligne
- Site Intranet pour le partage de fichiers
- Site Web destiné au public
- Installation et administration simples
- Antivirus et filtrage antispam
- Support de la communauté Microsoft
- Disponibilité à 99,9 % garantie financièrement
- Office Web Apps (Word, Excel, PowerPoint, et OneNote)
- Support en direct par téléphone 24h/24 et 7j/7
- Synchronisation Active Directory
- Prise en charge de la messagerie vocale hébergée
- Stockage du courrier électronique et archivage illimités

RÉSUMÉ



Created by Bharat
from the Noun Project

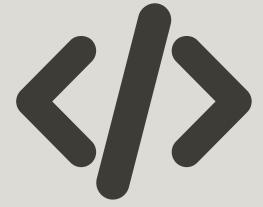
- Les besoins ne sont pas juste les fonctionnalités
 - Besoins non fonctionnels (qualités) sont souvent ignorés: convivialité, fiabilité, extensibilité, etc.
 - FURPS+ facilite la classification des besoins
- Définition des besoins est itérative dans le processus unifié

MODULE TDD

1. TDD - Développement piloté par les test - 15.01m
2. TDD en pratique

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

KATA TDD



- Kata TDD avec TypeScript
- Expérimenté avec SGB
 - npm run test
 - npm run watch
 - npm run test - -g “nom ou partie du nom”
 - <https://github.com/yvanross/log210-systeme-gestion-bordereau-node-express-ts>

TDD LABORATOIRE

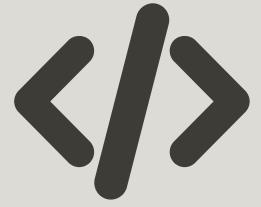


TDD 3 LAWS

1. You are not allowed to write any production code at all until you have written a failing unit test.

20 . 3

TDD LABORATOIRE



TDD 3 LAWS

1. You are not allowed to write any production code at all until you have written a failing unit test.
2. You are not allowed to write more of a unit test than is sufficient to fail and not compiling is failing.

20 . 3

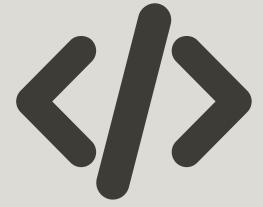
TDD LABORATOIRE



TDD 3 LAWS

1. You are not allowed to write any production code at all until you have written a failing unit test.
2. You are not allowed to write more of a unit test than is sufficient to fail and not compiling is failing.
3. You are not allowed to write more production code than is sufficient to pass the currently failing test. These three last gives us the perfect low-level documentation for a system code examples.

TDD



1. Kata TDD avec TypeScript

- Développer une fonction permettant d'extraire les mots uniques d'une phrase
- Développer une fonction permettant de faire l'intersetion des mots de deux phrases

MODULE GOF

1. Qu'est ce qu'un design pattern? - 14.25m
2. Adaptateur, Fabrique concrète, Singleton - 09.21m
3. Logique de tarification élaborée avec patron Stratégie et Composite - 18.57m
4. Actualisation interface usager - 06.21m
5. Révision Observateur - 08.47m
6. Découverte des patrons GOF - 10.23m
7. Recouvrement avec Proxy - 16.45m
8. Attention à la patternite
9. GRASP dans les GoF 07.27m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexité](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

21 . 1

QU'EST CE QU'UN DESIGN PATTERN?



Created by Jonathan Li
from the Noun Project

- Solution à un problème récurrent dans un contexte particulier

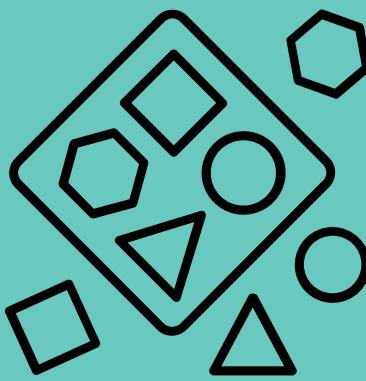
21 . 2

TROIS DÉFIS POUR BIEN UTILISER LES PATTERNS



Created by Jonathan Li
from the Noun Project

1. (Re)connaître le problème
2. Comprendre le patron (la solution)
3. Appliquer la solution



Created by Made by Made
from the Noun Project

1.(RE)CONNAÎTRE LE PROBLÈME



Created by Jonathan Li
from the Noun Project

- Comprendre le problème
- Exercice de « croyance » (on débute en OO)
- Certains problèmes sont plus « récurrents » que d'autres...
- Auteurs (GoF) avaient beaucoup d'expérience en OO

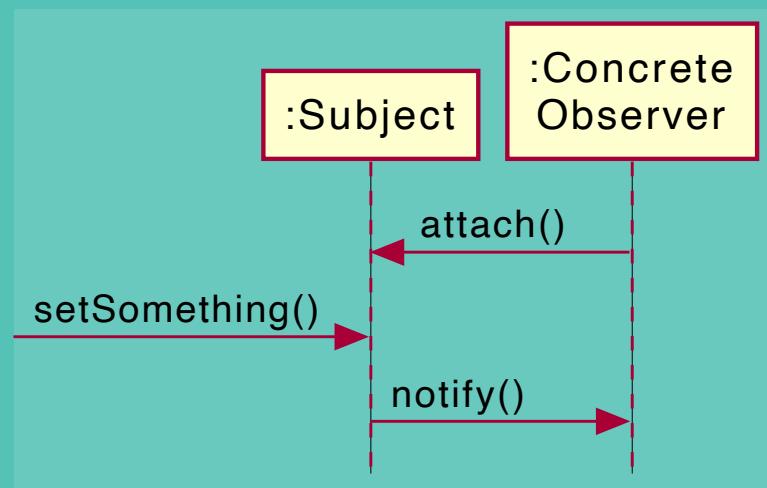
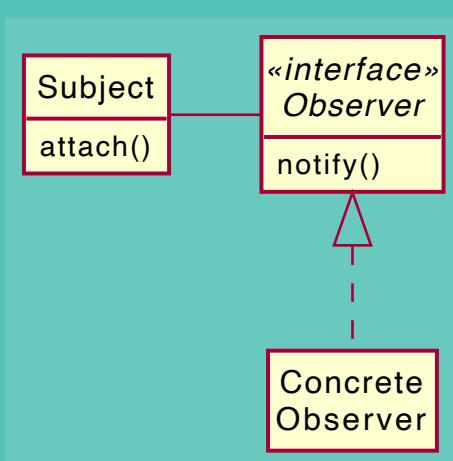


21 . 4

2.COMPRENDRE LE PATRON (LA SOLUTION)

Created by Jonathan Li
from the Noun Project

- abstractions
- notation UML est importante
- aspects statique et dynamique



3. APPLIQUER LA SOLUTION

- Étape concrète
- Mise en contexte particulier



Created by Jonathan Li
from the Noun Project

21 . 6

3. APPLIQUER LA SOLUTION



Created by Jonathan Li
from the Noun Project

Nom dans le patron

Vrai nom (Bouton Swing)

Subject

JButton

Observer

ActionListener

ConcreteObserver

la classe implémentant
l'interface ActionListener

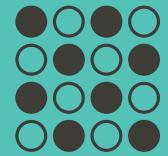
attach()

addActionListener()

notify()

actionPerformed()

PATRONS VUS DANS LE LIVRE DE LOG121



Created by Jonathan Li
from the Noun Project

- Adaptateur
- Commande
- Composite
- Décorateur
- Itérateur
- Méthode fabrique
- Observateur
- **Proxy**
- Singleton
- Stratégie
- Template
- Visiteur

PERSPECTIVE ZEN

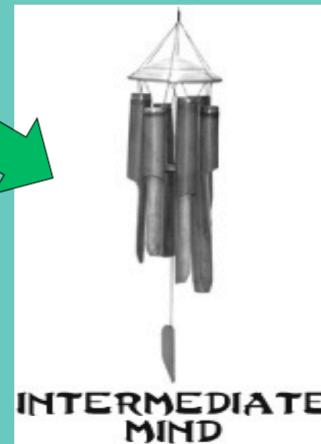


Created by Jonathan Li
from the Noom Project

Freeman, Elisabeth, Eric Freeman,
Bert Bates, Kathy Sierra, and Elisabeth
Robson. 2004. *Head First Design
Patterns*. 1st ed. O'Reilly Media.



« J'ai besoin d'un pattern pour
Bonjour, le Monde »



« Peut-être un singleton va-t-il
bien ici? »



« Un décorateur va naturellement
ici. »

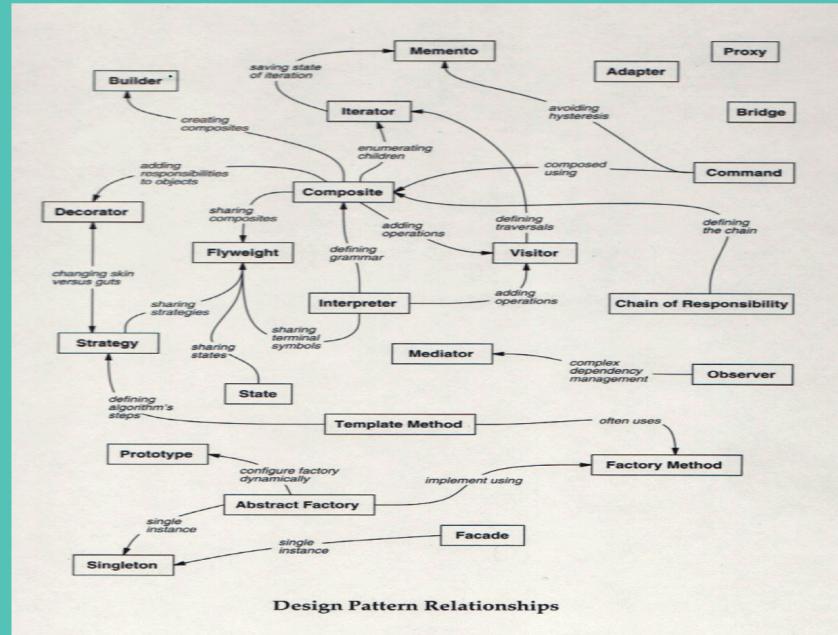


21 . 9

DESIGN PATTERN RELATIONSHIPS



Created by Jonathan Li
from the Noun Project



Ref: Design Patterns, Element of Reusable Object-oriented Software

RÉSUMÉ



Created by Jonathan Li
for the Noun Project

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate

21 . 11

RÉSUMÉ

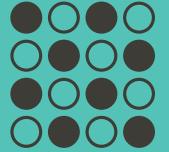


Created by Jonathan Li
for the Noun Project

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)

21 . 11

RÉSUMÉ



Created by Jonathan Li
for the Noam Project

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!

21 . 11

RÉSUMÉ



Created by Jonathan Li
for the Noam Project

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!
- Discuter de leurs avantages et leurs inconvénients avec des collègues

21 . 11

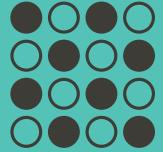
RÉSUMÉ



Created by Jonathan Li
from the Noam Project

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!
- Discuter de leurs avantages et leurs inconvénients avec des collègues
- Vérifier après l'application du pattern que les bénéfices sont vraiment là

RÉSUMÉ



Created by Jonathan Li
from the Noam Project

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!
- Discuter de leurs avantages et leurs inconvénients avec des collègues
- Vérifier après l'application du pattern que les bénéfices sont vraiment là
 - p.ex. il est plus facile d'étendre le logiciel

PATRON GOF LOG121



Created by Jonathan Li
from the Noun Project

- Décorateur
- Itérateur
- Commande
- Façade
- Singleton
- Template method
- Factory method
- Proxy
- Observateur
- Visiteur

21 . 12

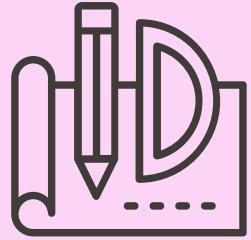
PATRONS GOF



Created by Jonathan Li
from the Noun Project

Quiz patron GOF

21 . 13



*PROBLÈME

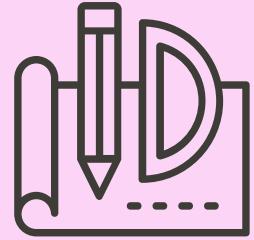
- POS NextGen doit supporter: (1) calcul des taxes, (2) autorisation de crédit, (3) comptabilité, (4) gestion des stocks
- Services trop compliqués pour le périmètre du projet → Services tiers « externes »
- Plusieurs fournisseurs de service (comme *TaxMaster* et *GoodAsGoldTaxPro*)
- L'API de chaque service est **distincte** et **immuable**

Patron GOF?

21 . 14

ADAPTATEUR (GOF)

*PROBLÈME



- POS NextGen doit supporter: (1) calcul des taxes, (2) autorisation de crédit, (3) comptabilité, (4) gestion des stocks
- Services trop compliqués pour le périmètre du projet → Services tiers « externes »
- Plusieurs fournisseurs de service (comme *TaxMaster* et *GoodAsGoldTaxPro*)
- L'API de chaque service est **distincte** et **immuable**

Patron GOF?

21 . 14

ADAPTATEUR (GOF)



Created by Jonathan Li
from the Noun Project

- Problème (suite)
 - On veut éviter ce genre de solution:

```
if [TaxMaster] {           L'API de chaque service est  
    taxes = tm.calculateTaxes(montant);   distincte et immuable.  
} elseif [GoodAsGold] {  
    taxes = gag.computeTaxes(montant);  
} else if [...] {          L'API de chaque service est  
    ...                         distincte et immuable.  
}
```

GRASP Polymorphisme

ADAPTATEUR POUR LE CALCUL DES TAXES

Created by Jonathan Li
from the Noun Project

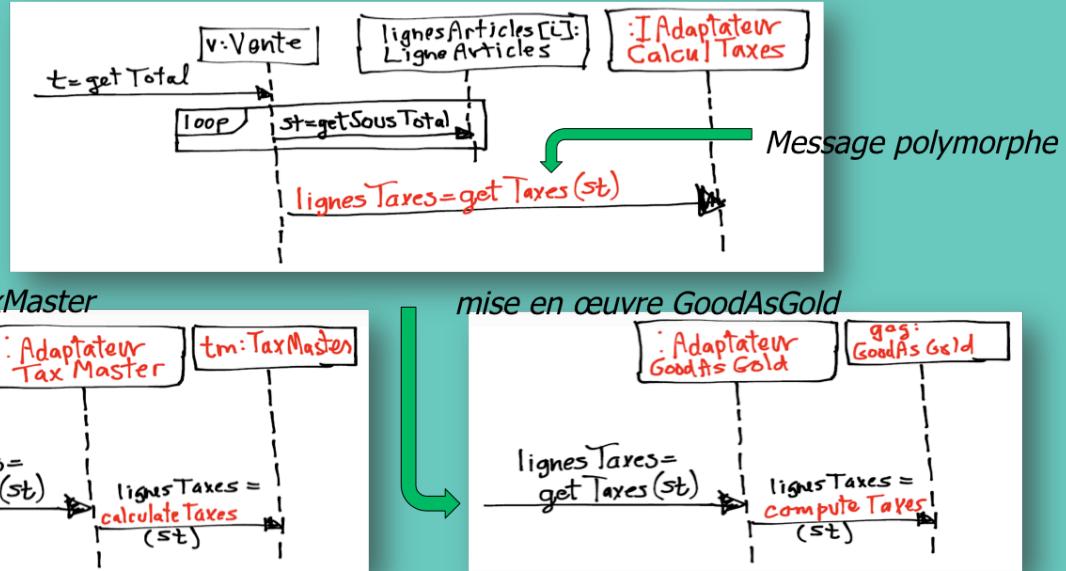


fig. F14.21



Created by Jonathan Li
from the Noun Project

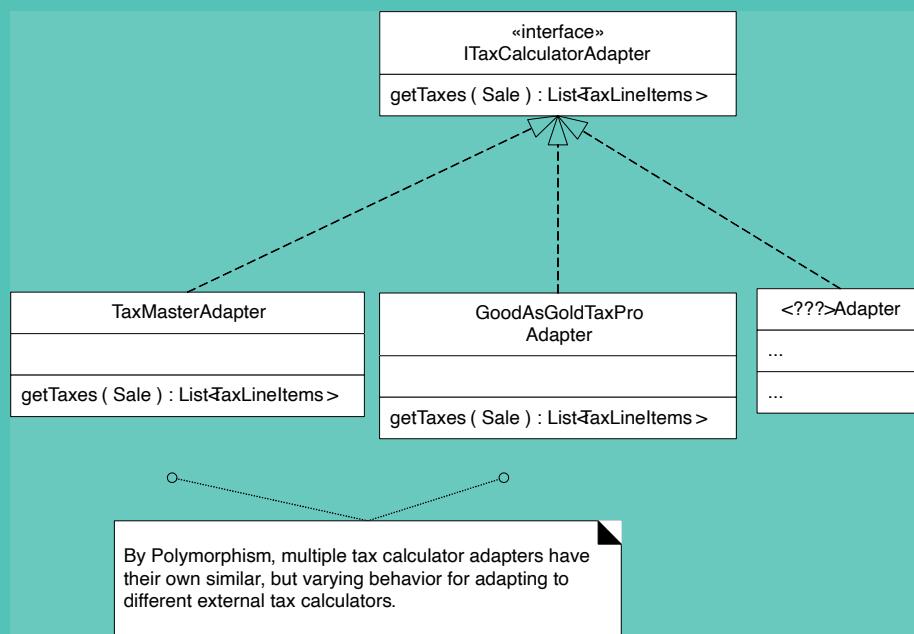


fig. F22.1/A25.1



Created by Jonathan Li
from the Noun Project

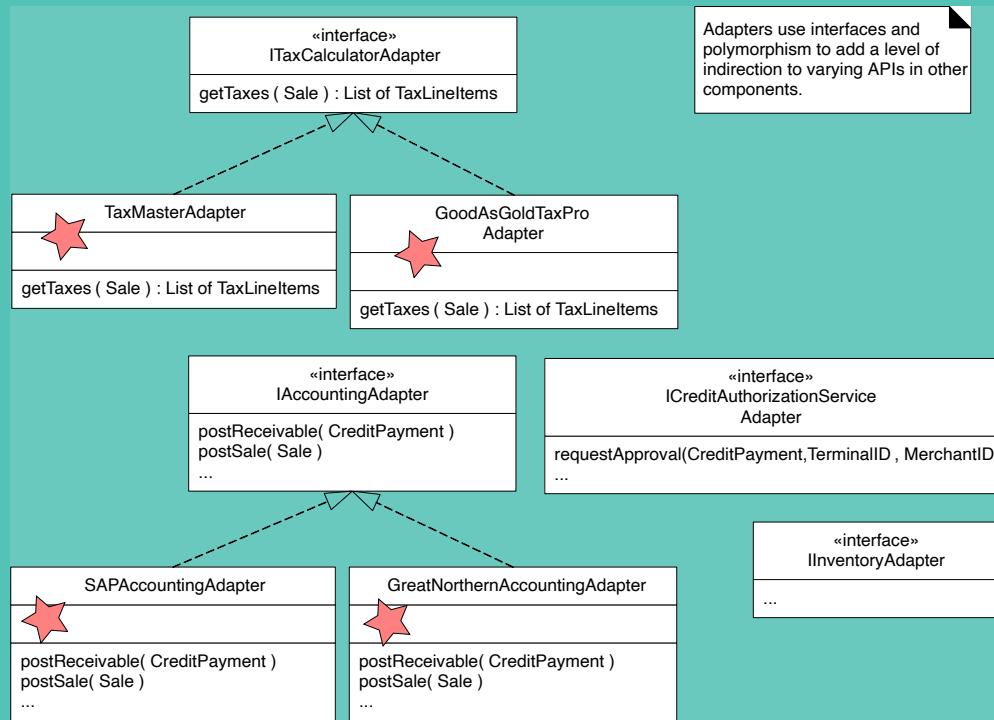


fig. F23.1

COMPTABILITÉ



Created by Jonathan Li
from the Noun Project

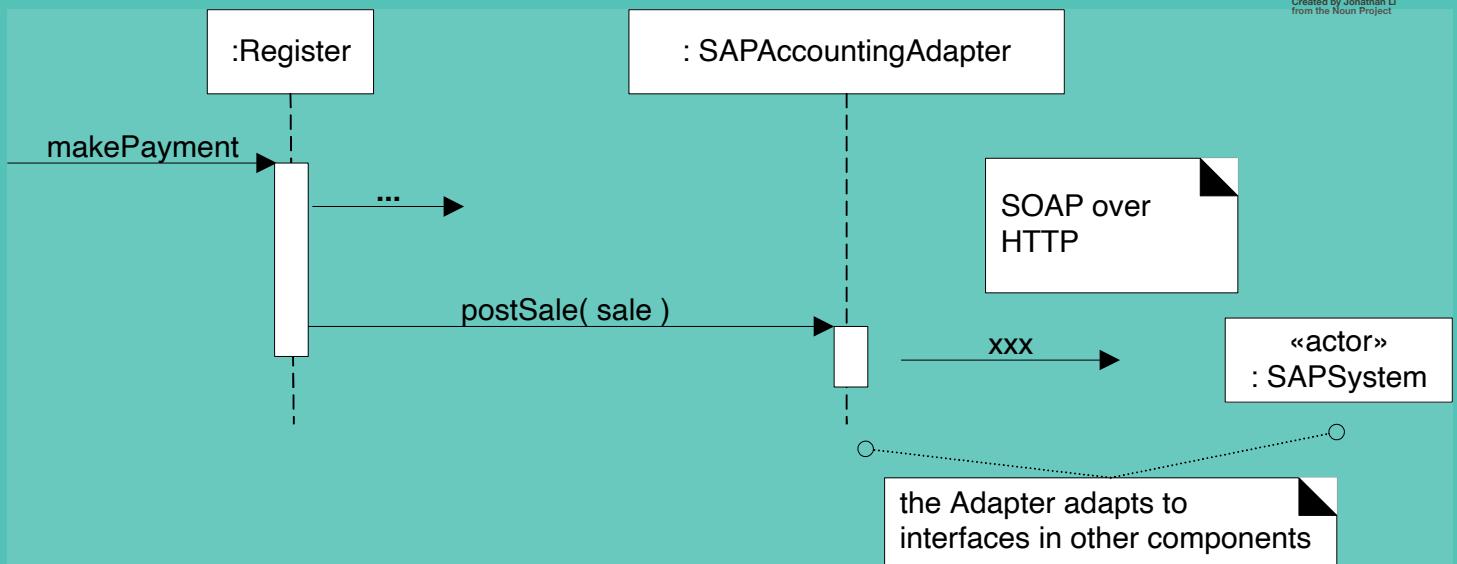


fig. F23.2

*GRASP DANS LE LABORATOIRE



Created by Jonathan Li
from the Noun Project

- SGB est un système externe alors avez vous utilisé un patron GOF pour que SGA se connecte a SGB.
- Si oui, quel patron et pourquoi
- Si non, pourquoi

*GRASP DANS LE LABORATOIRE



Created by Jonathan Li
from the Noun Project

- SGB est un système externe alors avez vous utilisé un patron GOF pour que SGA se connecte a SGB.
- Si oui, quel patron et pourquoi
- Si non, pourquoi
- Adaptateur? -> cache mémoire accès information

*GRASP DANS LE LABORATOIRE



Created by Jonathan Li
from the Noun Project

- SGB est un système externe alors avez vous utilisé un patron GOF pour que SGA se connecte a SGB.
- Si oui, quel patron et pourquoi
- Si non, pourquoi
- Adaptateur? -> cache mémoire accès information
- Proxy? Persistance pour écrire lorsque SGB n'est pas disponible

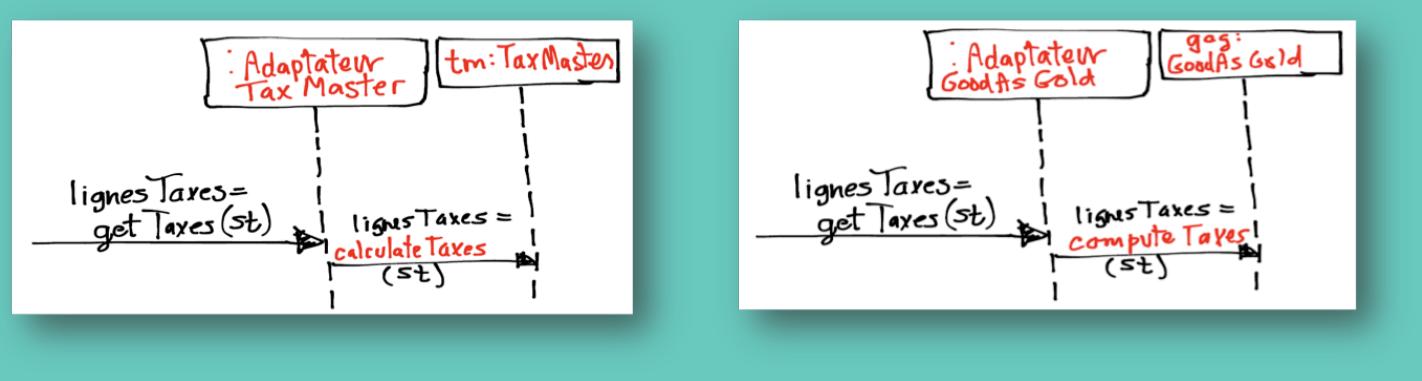
ADAPTATEUR - PROBLÈMES



Created by Jonathan Li
from the Noun Project

- Qui instancie les adaptateurs?
- Comment déterminer la classe d'adaptateur à instancier?

Adaptateur TaxMaster vs Adaptateur GoodAsGold





Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - Créer des objets
 - logique de création complexe, ou
 - séparer les responsabilités de créations (+ cohésion)
- Solution
 - Fabrication Pure nommé FabriqueConcrète pour gérer la création

FABRIQUE CONCRÈTE (PAS GOF)



Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - Créer des objets
 - logique de création complexe, ou
 - séparer les responsabilités de créations (+ cohésion)
- Solution
 - Fabrication Pure nommé FabriqueConcrète pour gérer la création

FABRIQUE CONCRÈTE



Created by Jonathan Li
from the Noun Project

FabriqueDeServices

```
adaptateurCompta : IAdaptateurCompta
adaptateurStock : IAdaptateurStock
adaptateurCalculateurTaxes : IAdaptateurCalculateurTaxes.
```

```
getAdaptateurCompta() : IAdaptateurCompta
getAdaptateurStock() : IAdaptateurStock
getAdaptateurCalculateurTaxes() : IAdaptateurCalculateurTaxes
...
```

notez que les méthodes d'une Fabrique concrète retournent des objets typés interface (et non typés classe), afin de pouvoir retourner n'importe quelle implémentation de l'interface

```
if( adaptateurCalculateurTaxes ==null )
{
    // une approche réflexive ou pilotée par les données pour trouver la bonne classe :
    // lire dans une propriété externe

    String nomClasse = System.getProperty( "calculatortaxes.classe.nom" );
    adaptateurCalculateurTaxes = (IAdaptateurCalculateurTaxes) Class.forName(nomClasse).newInstance();

}
return adaptateurCalculateurTaxes ;
```

Documentation `System.getProperty()`

*PROBLÈME SUIVANT...



Created by Jonathan Li
from the Noun Project

- Qui instancie la Fabrique elle-même?
- Comment y accéder?
- Indices
 - besoin de seulement une instance de chaque Fabrique
 - méthodes de la Fabrique appelées à partir d'endroits différents dans le code
 - problème de visibilité

*PROBLÈME SUIVANT...

PATRON SINGLETON



Created by Jonathan Li
from the Noun Project

- Qui instancie la Fabrique elle-même?
- Comment y accéder?
- Indices
 - besoin de seulement une instance de chaque Fabrique
 - méthodes de la Fabrique appelées à partir d'endroits différents dans le code
 - problème de visibilité

SINGLETON



Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - Une seule instance est permise: le « singleton »
 - besoin d'un point d'accès global et unique
- Solution
 - méthode statique qui retourne le singleton

SINGLETON



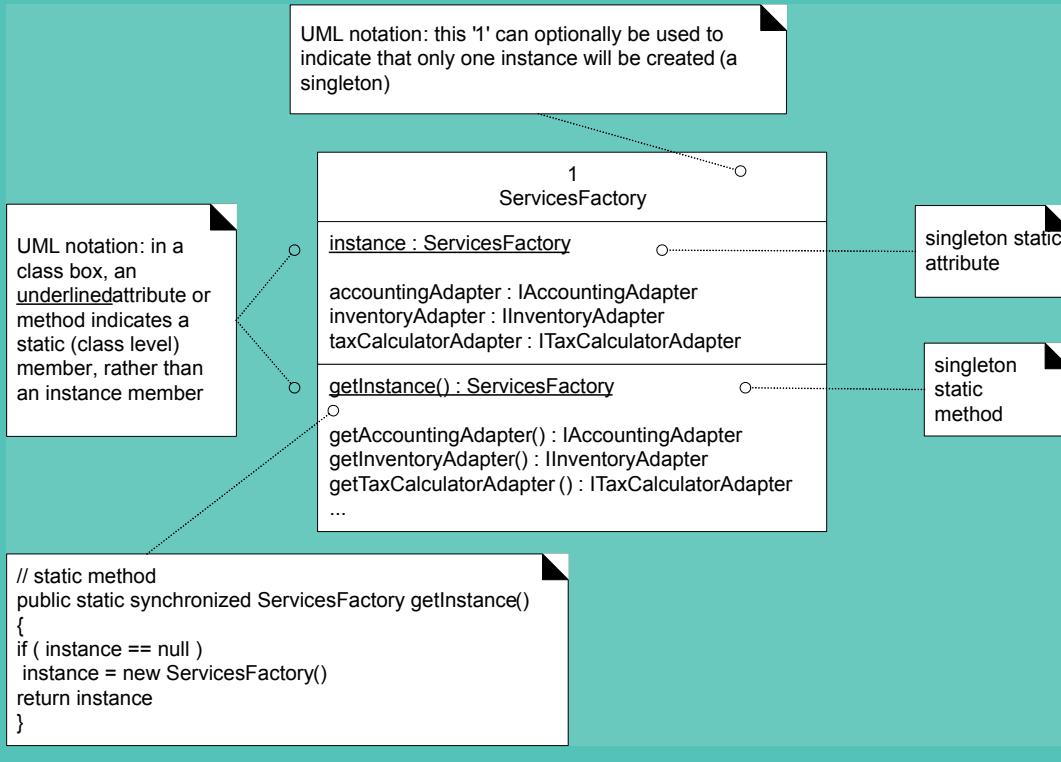
Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - Une seule instance est permise: le « singleton »
 - besoin d'un point d'accès global et unique
 - Solution
 - méthode statique qui retourne le singleton
- ⚠ IL EST DIFFICILE DE TESTER UN SINGLETON**

SINGLETON



Created by Jonathan Li
from the Noun Project



SINGLETON - IMPLÉMENTATION JAVA

Created by Jonathan Li
from the Noun Project

- Initialisation différée
- `getInstance()` est souvent fréquemment appelé
- Avec plusieurs fils d'exécution (threads), l'étape de création d'une logique d'initialisation différée (lazy initialization) est critique
 - nécessite un contrôle de la concurrence des threads (p.ex. une méthode atomique)

```
public static synchronized FabriqueDeServices getInstance () {  
    if (instance == null ) {  
        // section critique pour une application multithread  
        instance = new FabriqueDeServices();  
    }  
    return instance;  
}
```

référence: <http://www.cs.wustl.edu/~schmidt/PDF/monitor.pdf>

SINGLETON - IMPLÉMENTATION JAVA



```
public class FabriqueDeServices {  
    // initialisation immédiate  
    private static FabriqueDeServices instance =  
        new FabriqueDeServices();  
    public static FabriqueDeServices getInstance () {  
        return instance;  
    } // autres méthodes...  
}
```

- Initialisation immédiate
- Quelle initialisation est préférée?
- Initialisation différée a plusieurs avantages:
 - si l'objet Singleton n'est jamais utilisé, on évite un travail de création inutile
 - getInstance() comprend parfois une logique de création complexe et conditionnelle (difficile à mettre dans une initialisation immédiate)

SINGLETON



Created by Jonathan Li
from the Noun Project

- Message implicite « getInstance »
- Pas besoin de l'écrire dans la dynamique

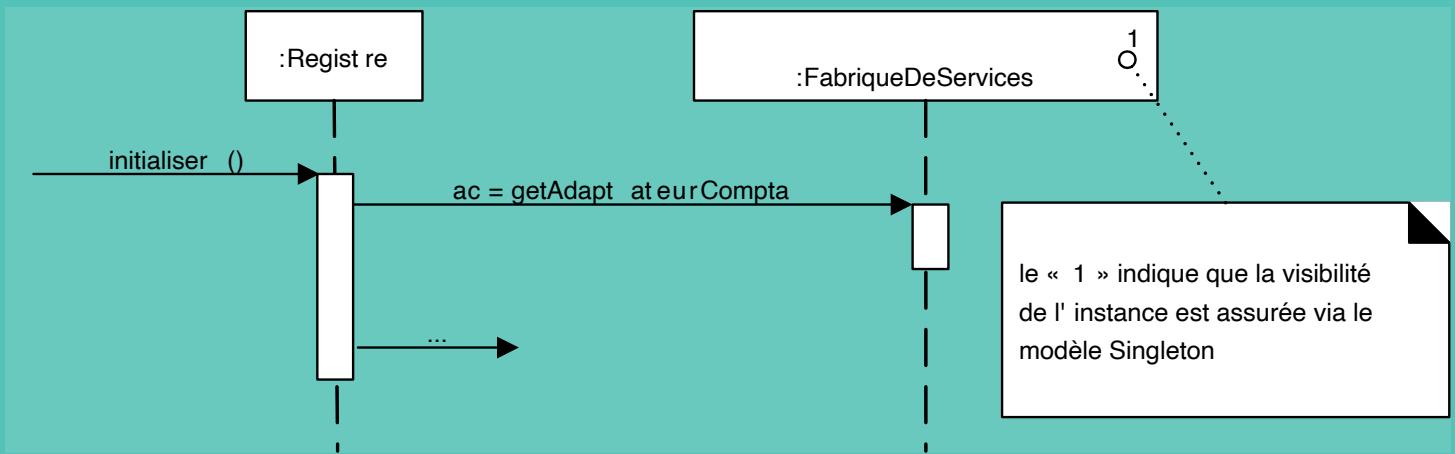
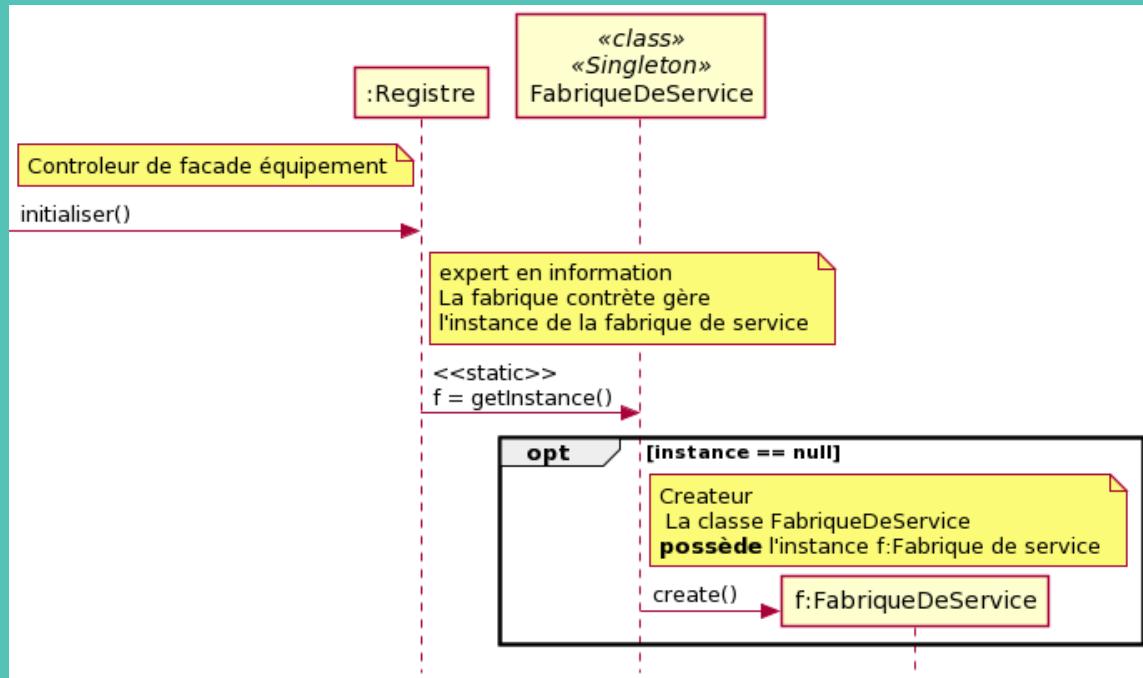


fig. F23.7

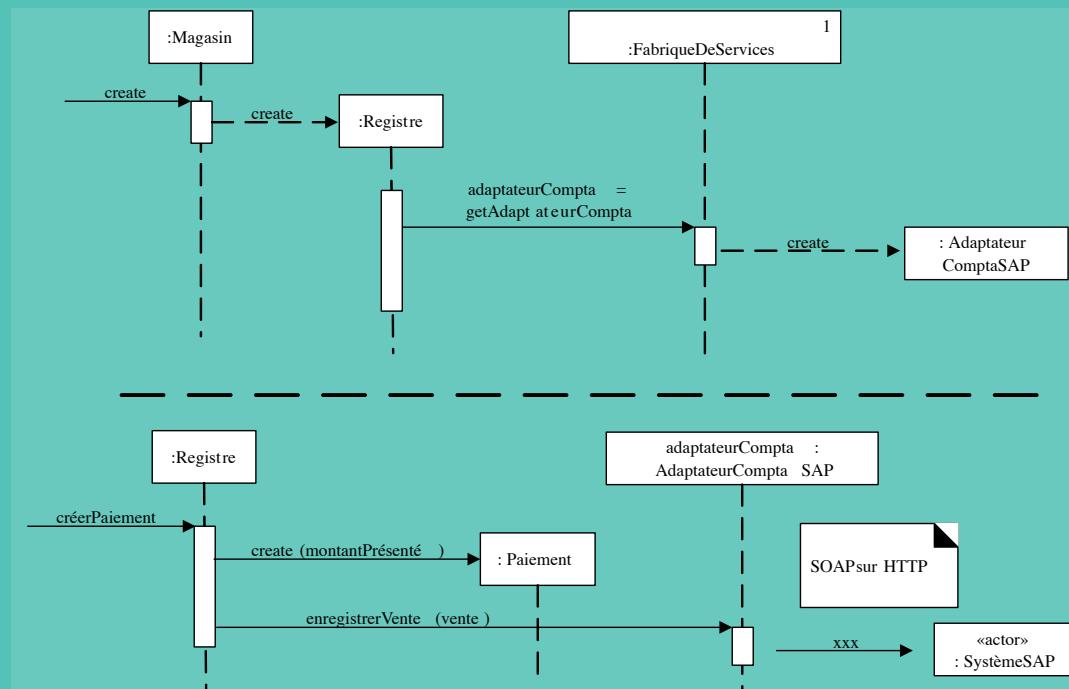


SINGLETON

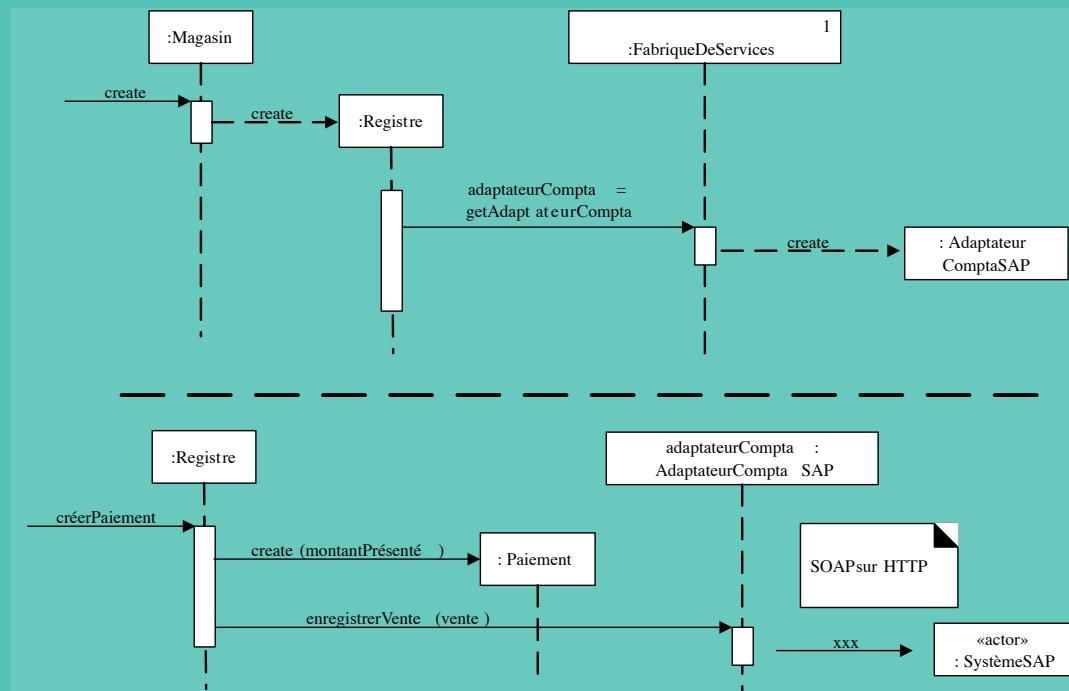
FAIRE LE DS DE LA SOLUTION PRÉCÉDENTE EN INCLUANT LA DYNAMIQUE DE GETINSTANCE



*CONCLUSION SUR LES SERVICES EXTERNES.



*CONCLUSION SUR LES SERVICES EXTERNES.



Quel message est polymorphe?

LOGIQUE DE TARIFICATION ÉLABORÉE



Created by Jonathan Li
from the Noun Project

- Support pour la logique des promotions
 - Réductions, soldes, rabais, etc.
- « Stratégie de tarification » peut varier
 - Règle, algorithme, politique, etc.
- Exemples
 - réduction de 10% sur toute les ventes pour une période
 - réduction fixe de 10 dollars pour tout achat d'un montant supérieur à 200\$
 - etc.
- Il ne faut pas reprogrammer le logiciel pour chaque nouvelle promotion
- Comment faire un bon design face à cette exigence?

Patron GOF?



Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - algorithmes ou politiques sont variables mais se ressemblent (politiques de tarification)
 - ils doivent pouvoir évoluer
- Solution
 - définir les algorithmes/politiques/stratégies dans une classe séparée, avec une interface commune.

Patron GOF?

STRATÉGIE

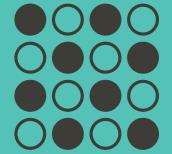


Created by Jonathan Li
from the Noun Project

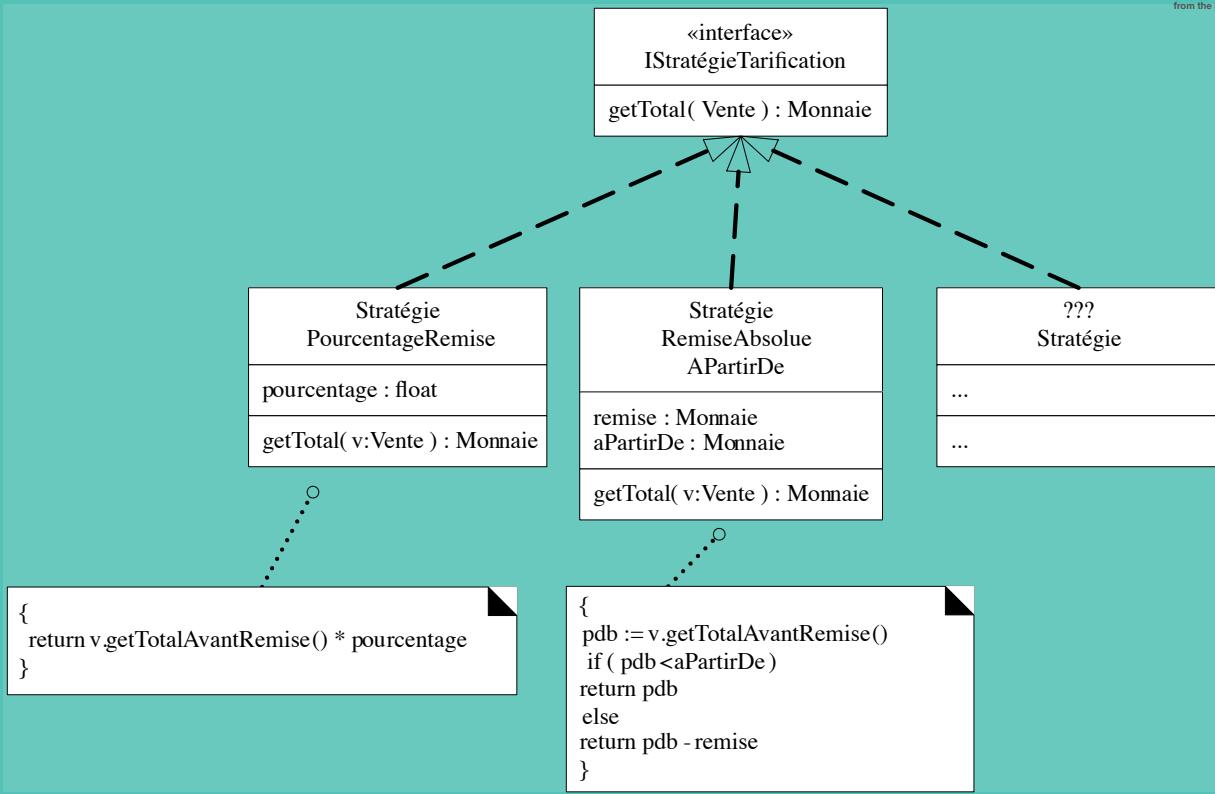
- Contexte / Problème
 - algorithmes ou politiques sont variables mais se ressemblent (politiques de tarification)
 - ils doivent pouvoir évoluer
- Solution
 - définir les algorithmes/politiques/stratégies dans une classe séparée, avec une interface commune.

Patron GOF?

STRATÉGIES DES PRIX

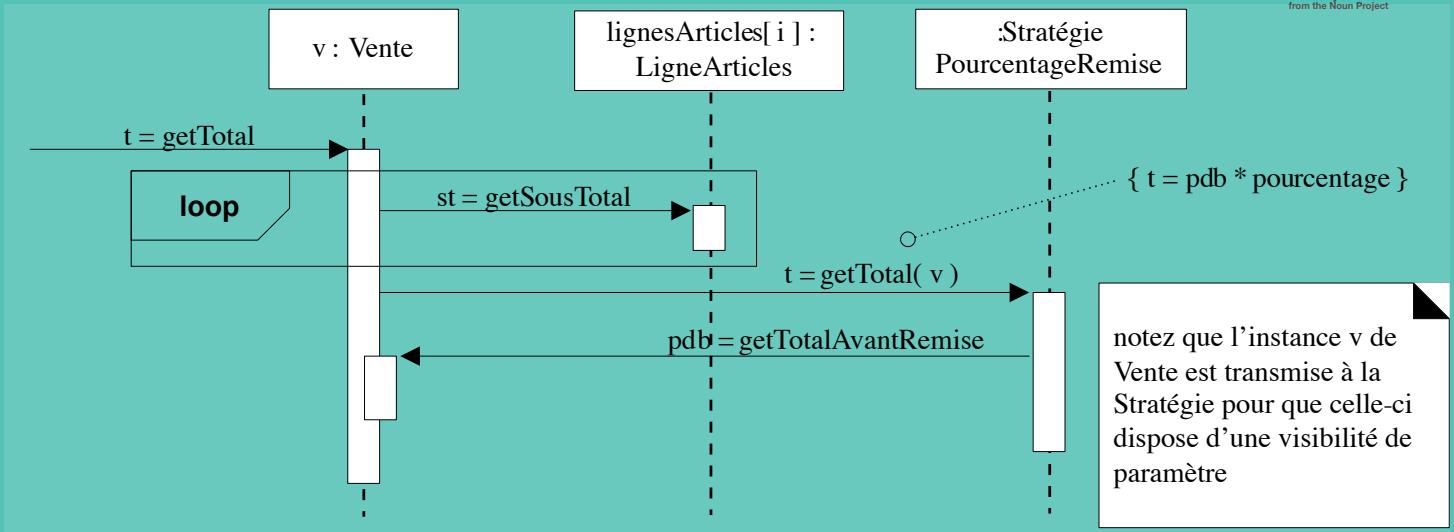


Created by Jonathan Li
from the Noun Project



STRATÉGIE ET OBJET CONTEXT (VENTE)

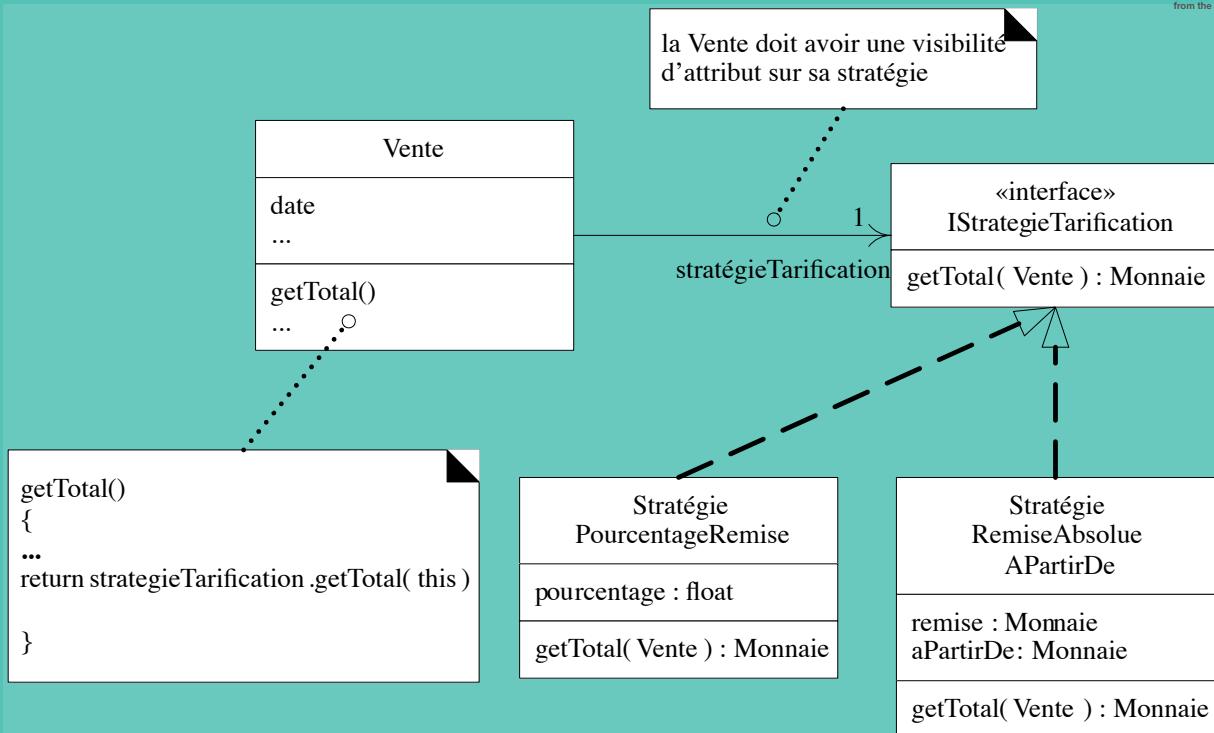
Created by Jonathan Li
from the Noun Project



L'OBJET CONTEXTE (VENTE)



Created by Jonathan Li
from the Noun Project



CRÉER UNE STRATÉGIE

Created by Jonathan Li
from the Noun Project

1

FabriqueDeStratégiesTarification

instance : FabriqueDeStratégiesTarificationgetInstance() : FabriqueDeStratégiesTarification

getStratégieTarification() : IStratégieTarification

getStratégieTarificationSenior() : IStratégieTarification

...

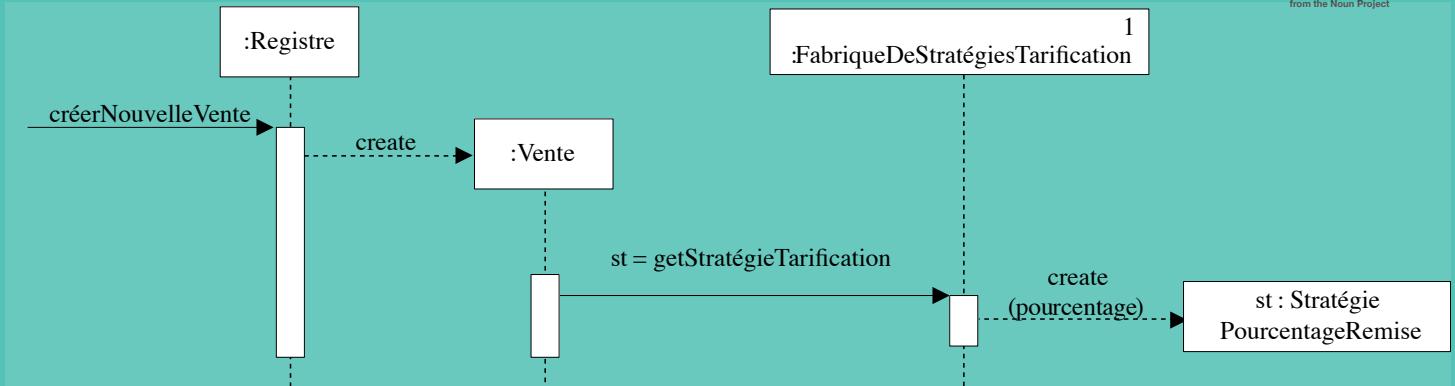
{

```
String nomClasse= System.getProperty( "strategietarification.classe.nom" );
strategie = (IStratégieTarification) Class.forName( nomClasse ).newInstance();
return strategie;
}
```

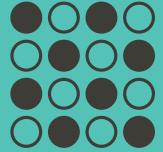
CRÉER UNE STRATÉGIE



Created by Jonathan Li
from the Noun Project



EN RÉSUMÉ



Created by Jonathan Li
from the Noun Project

- *Stratégie et Fabrique concrète assurent la Protection des variations dues aux changements dynamiques de politique tarifaire*
- *Stratégie → Polymorphisme (GRASP)*
- Fabriques sont souvent utilisées pour créer les stratégies pour autoriser des algorithmes insérables



Created by Jonathan Li
from the Noun Project

CONFLITS DUS AUX POLITIQUES TARIFAIRES

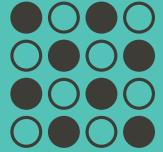
- Politiques de prix pour aujourd’hui (lundi)
 - 20% de rabais pour les personnes du troisième âge
 - 15% de rabais sur un achat supérieur à 400\$
 - 50 dollars de rabais pour sur un achat supérieur à 500\$ (ce lundi seulement)
 - Si on achète un carton de thé Darjeeling, il y a un rabais de 15% sur tout
- Comment résoudre les cas conflictuels?
 - Une dame de 77 ans qui est aussi client privilégié achète une caisse de Darjeeling et dépense 600\$

Patron GOF?

21 . 40

COMPOSITE

CONFLITS DUS AUX POLITIQUES TARIFAIRES



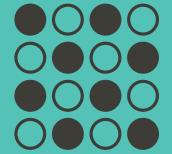
Created by Jonathan Li
from the Noun Project

- Politiques de prix pour aujourd’hui (lundi)
 - 20% de rabais pour les personnes du troisième âge
 - 15% de rabais sur un achat supérieur à 400\$
 - 50 dollars de rabais pour sur un achat supérieur à 500\$ (ce lundi seulement)
 - Si on achète un carton de thé Darjeeling, il y a un rabais de 15% sur tout
- Comment résoudre les cas conflictuels?
 - Une dame de 77 ans qui est aussi client privilégié achète une caisse de Darjeeling et dépense 600\$

Patron GOF?

21 . 40

FACTEURS DES STRATÉGIES



Created by Jonathan Li
from the Noun Project

- Période du temps (lundi)

FACTEURS DES STRATÉGIES



Created by Jonathan Li
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)

FACTEURS DES STRATÉGIES



Created by Jonathan Li
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)

FACTEURS DES STRATÉGIES



Created by Jonathan Li
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits

FACTEURS DES STRATÉGIES



Created by Jonathan Li
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
 - Prix le plus bas

FACTEURS DES STRATÉGIES



Created by Jonathan Li
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
 - Prix le plus bas
 - Prix le plus haut

FACTEURS DES STRATÉGIES



Created by Jonathan Li
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
 - Prix le plus bas
 - Prix le plus haut
 - Etc.



Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - parfois on traite un seul objet
 - atomique
 - parfois on traite un groupe
 - composition d'objets
 - les traiter de la même façon
- Solution
 - définir des classes pour les objets
 - composites
 - atomiques
- implémentent la même interface

Patron GOF?

COMPOSITE



Created by Jonathan Li
from the Noun Project

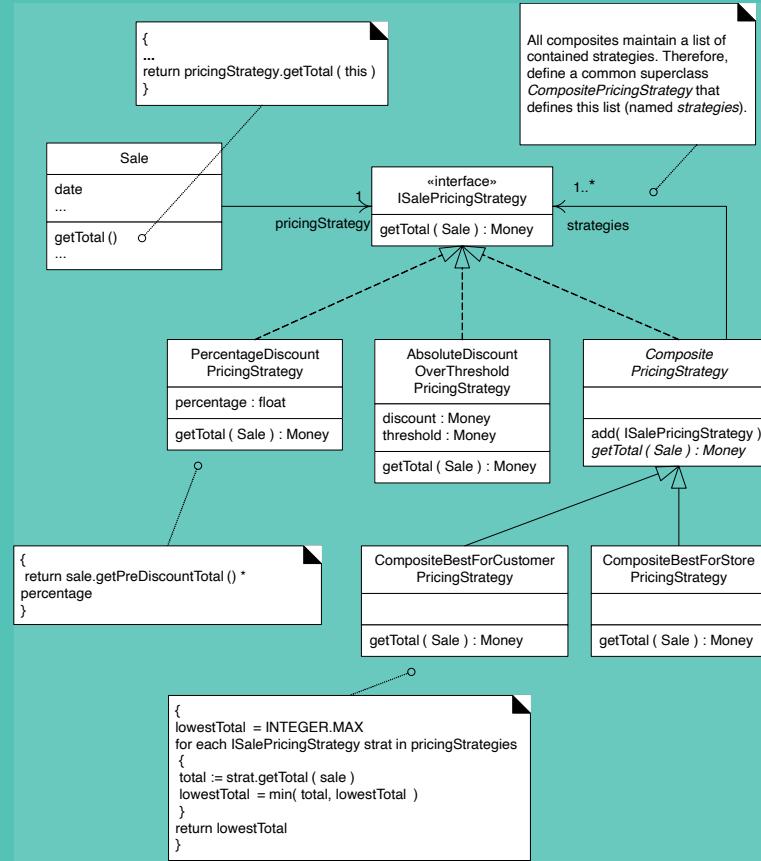
- Contexte / Problème
 - parfois on traite un seul objet
 - atomique
 - parfois on traite un groupe
 - composition d'objets
 - les traiter de la même façon
- Solution
 - définir des classes pour les objets
 - composites
 - atomiques
- implémentent la même interface

Patron GOF?



Created by Jonathan Li
from the Noun Project

COMPOSITE POS

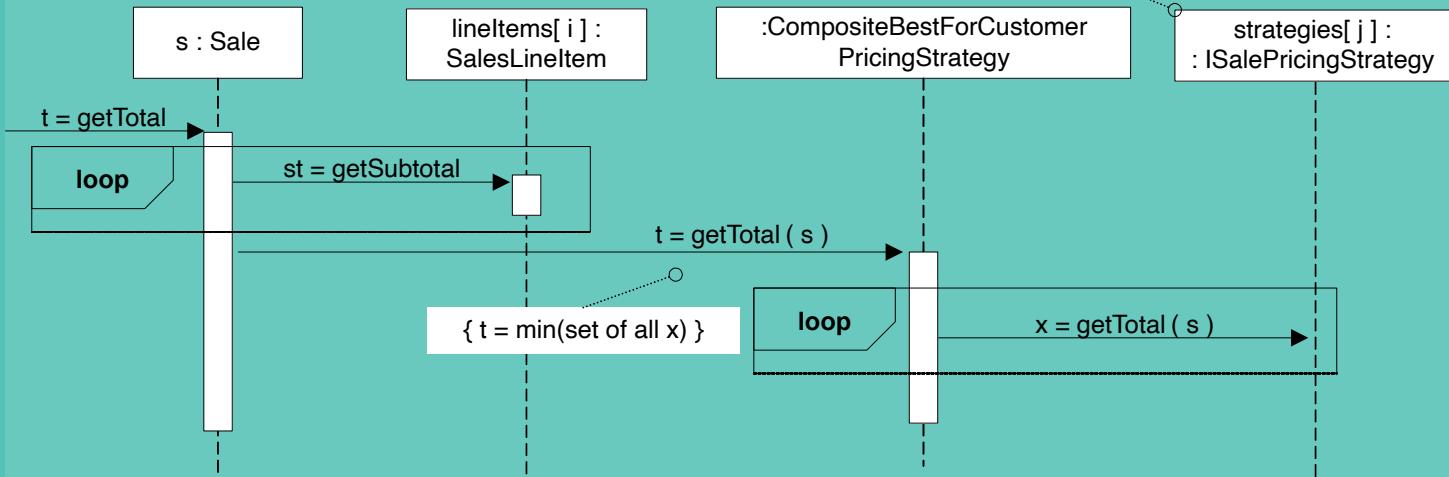


COMPOSITE - COLLABORATION



Created by Jonathan Li

UML: `ISalePricingStrategy` is an interface, not a class; this is the way in UML 2 to indicate an object of an unknown class, but that implements this interface

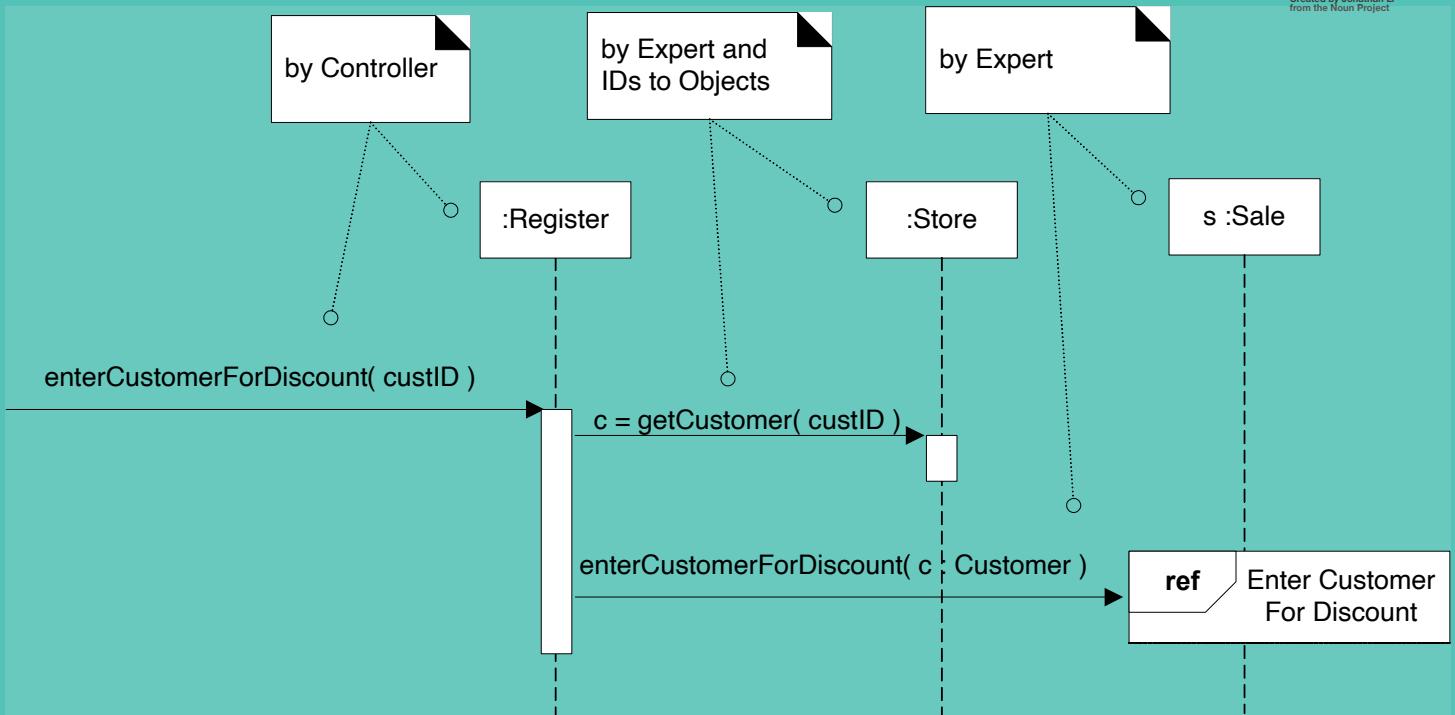


the `Sale` object treats a Composite Strategy that contains other strategies just like any other `ISalePricingStrategy`

RÉALISATION DE CAS D'UTILISATION

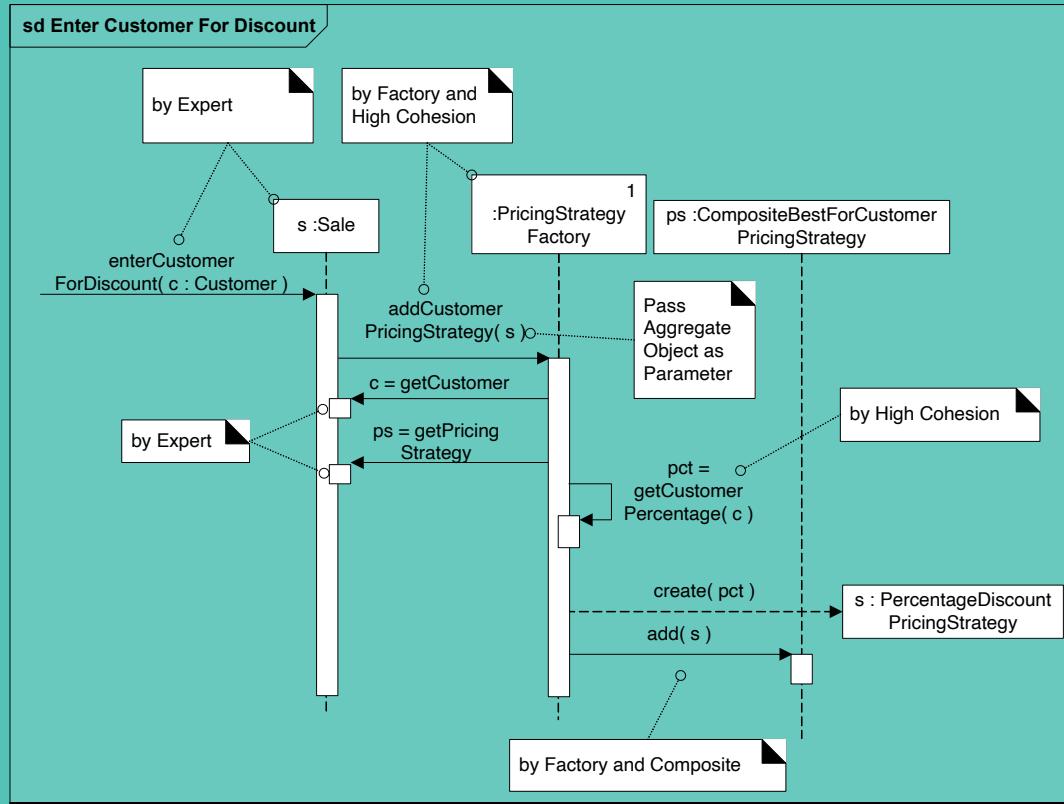


Created by Jonathan Li
from the Noun Project



RÉALISATION DE CAS D'UTILISATION

Created by Jonathan Li
from the Noun Project



RÈGLES D'AFFAIRES



Created by Jonathan Li
from the Noun Project

- Règles d'affaires personnalisables
- À certains points dans les scénarios
 - e.g., makeNewSale, enterItem
- Exemple : Paiement par bon de cadeau
 - limite l'utilisation d'un bon à un seul item
 - il n'y a pas de monnaie à rendre au client
 - etc.

IMPLEMENTATION DES RÈGLES



Created by Jonathan Li
from the Noun Project

- Implémentation des règles est inconnue
 - et peut être extensible
- Voudrait permettre
 - utilisation de plusieurs méthodes
 - patron stratégie/composite
 - interpréteurs de règles « open source »
 - interpréteurs de règles propriétaires (COTS)



Created by Jonathan Li
The Noun Project

ACTUALISATION

Goal: When the total of the sale changes, refresh the display with the new value

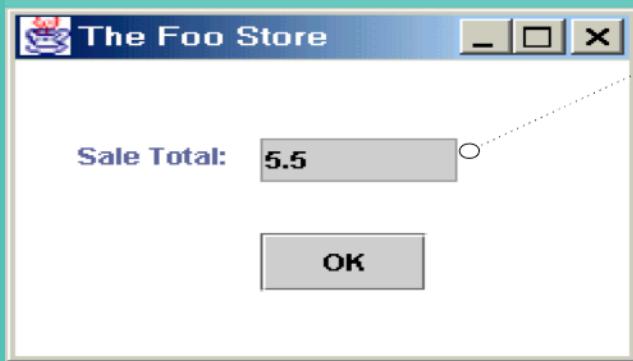


fig. F26.21

SOLUTION POSSIBLE...



Created by Jonathan Li
from the Noun Project

- Total de Sale est changé
 - envoie un message à la fenêtre GUI
 - lui demande de se mettre à jour
- Cette solution n'est pas recommandée
 - Sale ne devrait pas connaître le GUI
 - c'est une dépendance vers une classe dont le code risque de changer (la GUI est peu stable)
- Par exemple
 - si la classe Sale dépend de Swing
 - alors il est difficile de changer l'interface



Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - objet observateur
 - réagit selon les changements du sujet
 - objet sujet
 - annonce un changement
- minimiser le couplage du sujet vers l'observateur

OBSERVATEUR



Created by Jonathan Li
from the Noun Project

- Contexte / Problème
 - objet observateur
 - réagit selon les changements du sujet
 - objet sujet
 - annonce un changement
- minimiser le couplage du sujet vers l'observateur

OBSERVATEUR

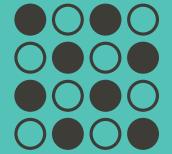


Created by Jonathan Li
from the Noun Project

- Solution

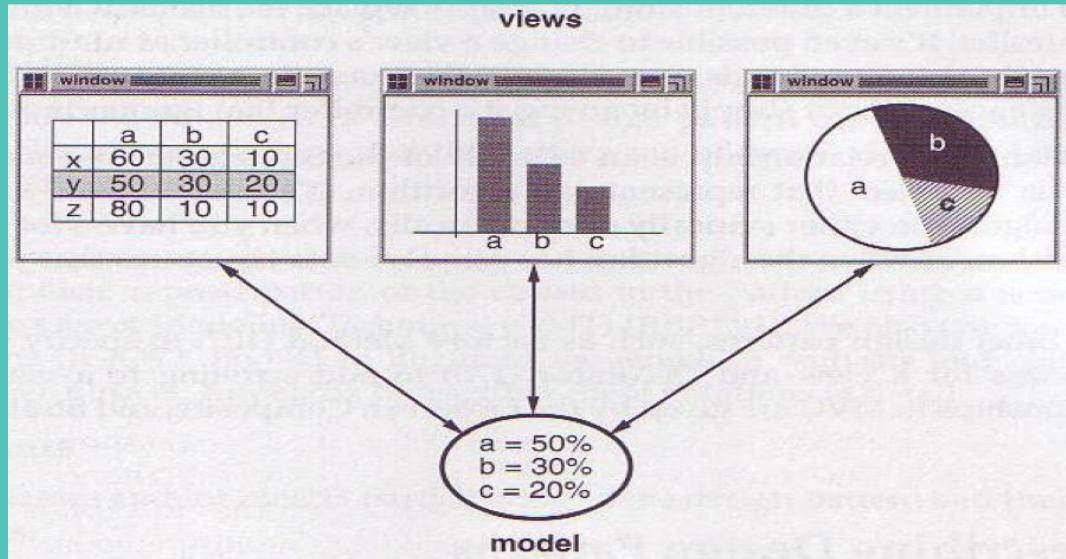
- interface observateur
- les observateurs s'inscrivent au sujet
- le sujet ne connaît pas les observateurs particuliers
 - seulement des objets implémentant l'interface
- le sujet envoie un message aux observateurs
 - lorsqu'un certain changement (événement) survient

OBSERVATEUR



Created by Jonathan Li
from the Noun Project

Plusieurs observateurs, un sujet



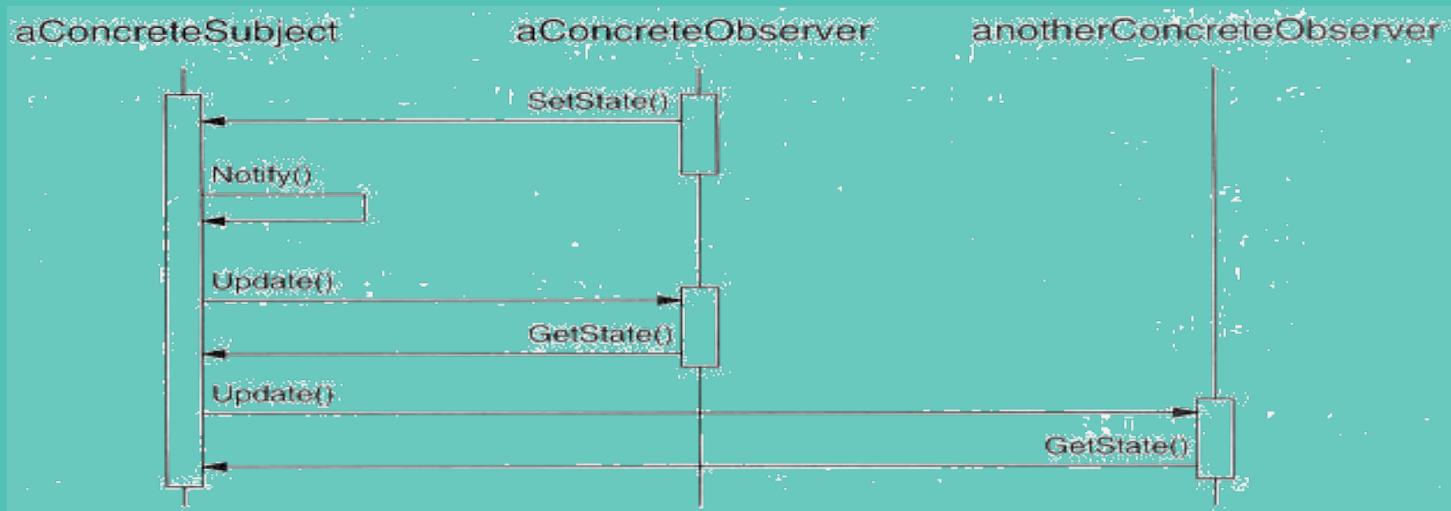
ref: Design Patterns, Gamma, Helm, Johnson & Vlissides, 1995

OBSERVATEUR



Created by Jonathan Li
from the Noun Project

Communication



ref: Design Patterns, Gamma, Helm, Johnson & Vlissides, 1995



Created by Jonathan Li
from the Noun Project

SOLUTION : OBSERVATEUR

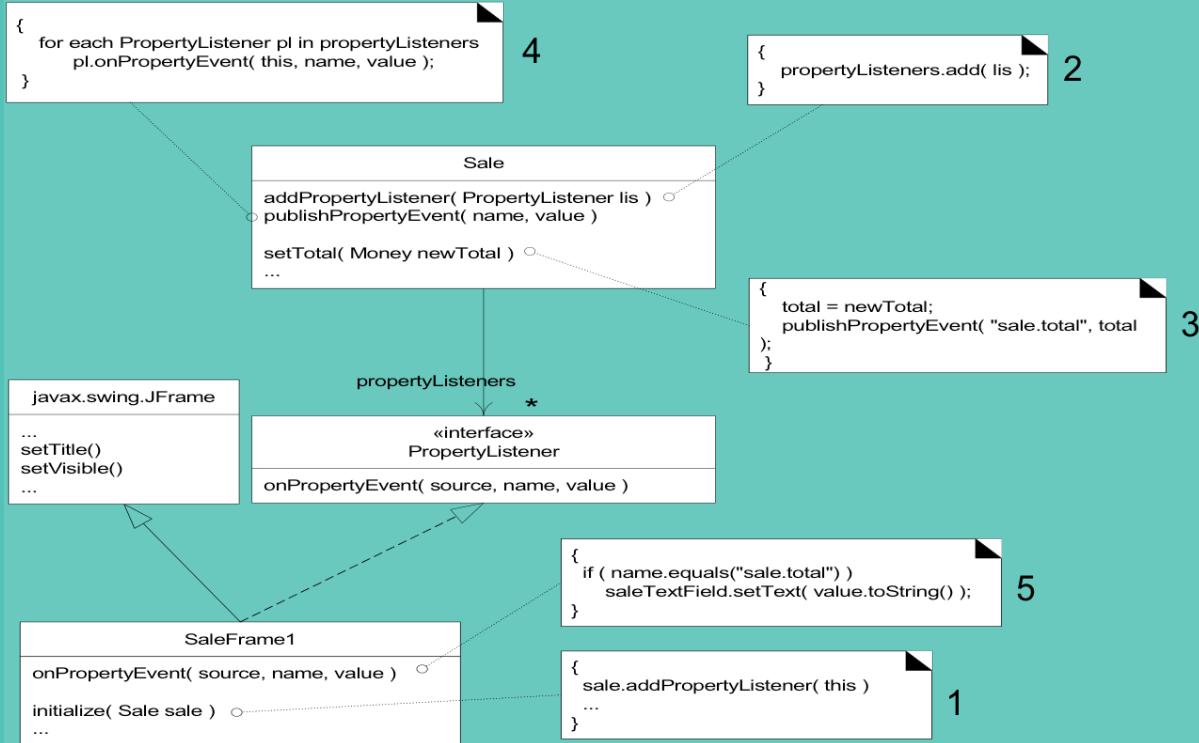


fig. F26.22

EXERCICE OBSERVATEUR



Created by Jonathan Li
from the Noun Project

- Rappeler le patron Observateur
- Réviser le diagramme de structure
- Réaliser les diagrammes de comportement
 - Communication
 - Activité
 - Séquence

SOLUTION : OBSERVATEUR STRUCTURE

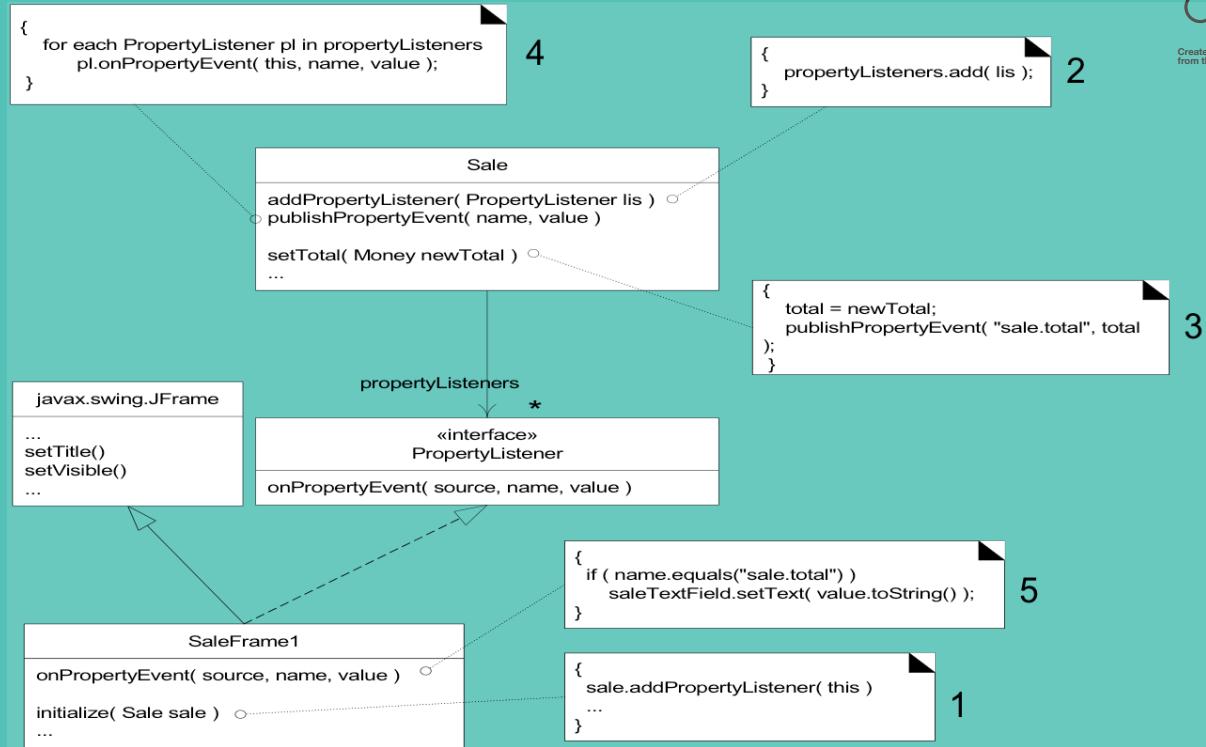
Created by Jonathan Li
from the Noun Project

fig. F26.22

OBSERVATEUR COMMUNICATION

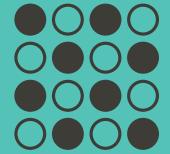


Created by Jonathan Li
from the Noun Project

Voir la note pour la solution

21 . 58

OBSERVATEUR ACTIVITÉ

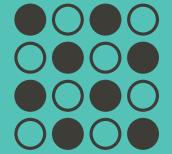


Created by Jonathan Li
from the Noun Project

Voir la note pour la solution

21 . 59

OBSERVATEUR SÉQUENCE



Created by Jonathan Li
from the Noun Project

Voir la note pour la solution

21 . 60

*DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION



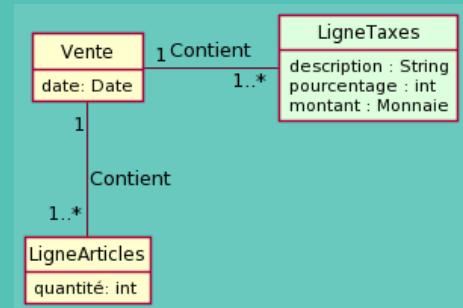
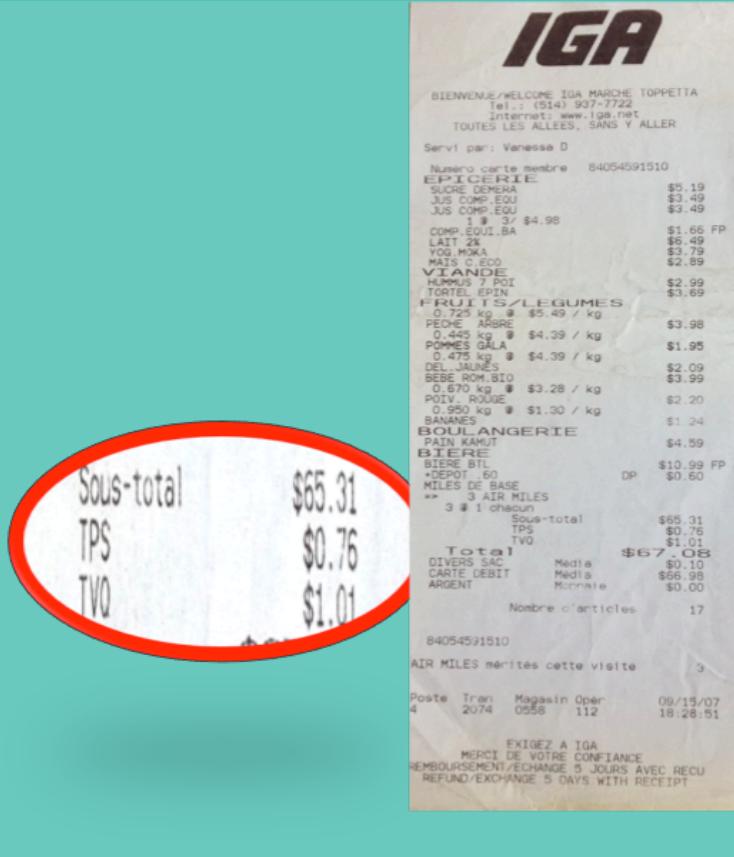
Created by Jonathan Li
from the Noun Project



*DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION



Created by Jonathan Li
from the Noun Project



DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION



- Calculateurs de taxes retournent LigneTaxes
 - pas envisagé lors de l'analyse initiale
- Concept dans le modèle du domaine
 - mettre à jour le modèle du domaine
 - mettre à jour le glossaire
- Avec l'expérience ceci devient un réflexe
 - travail d'un ingénieur (vs technicien ou codeur)
 - nécessaire dans un processus itératif

ATTRIBUTS DE QUALITÉ

- Appliquer les GoF et GRASP dans les RDCU pour traiter les attributs de qualité (exigences URPS)
 - NextGen
 - basculement sur les services locaux,
 - gestion du terminal POS et
 - autorisations de crédit
 - Monopoly
 - différentes cases
 - achat de propriétés
 - paiement de loyers

EXIGENCES NEXTGEN



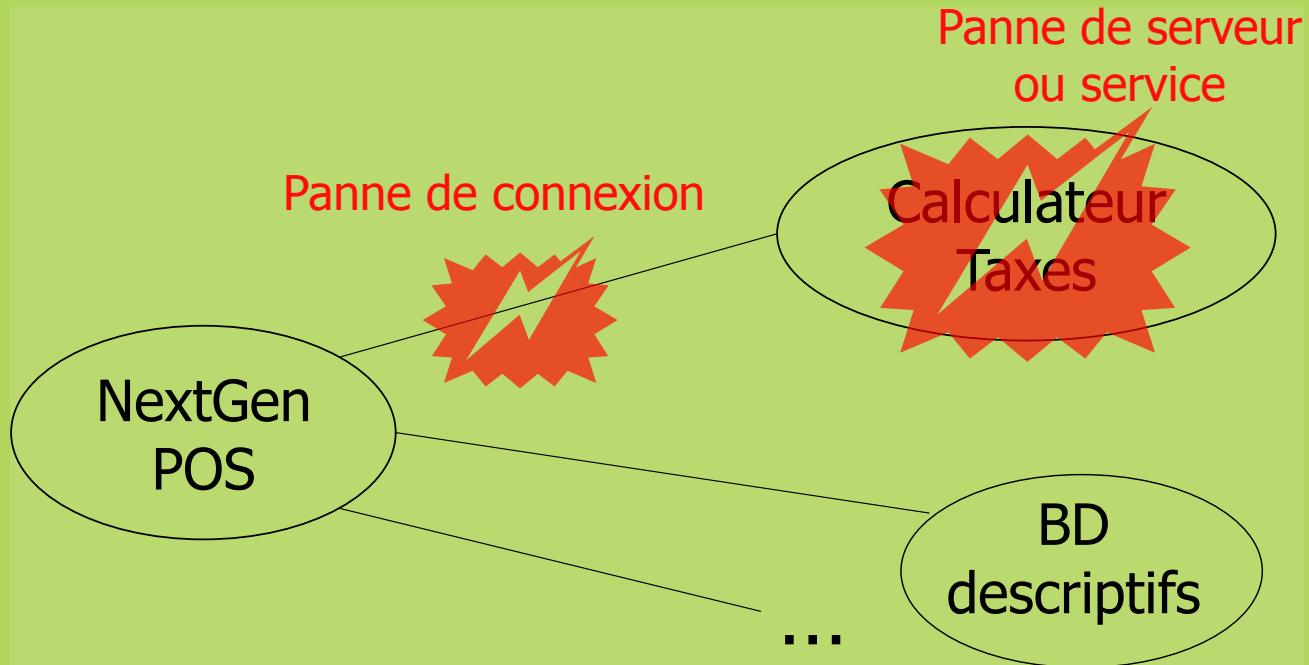
Created by Bharat
from the Noun Project

- En cas d'échec d'un service distant, le système doit continuer à fonctionner (tolérance aux pannes, **fiabilité**)
- Mise en cache locale (**performance**)
- Prise en charge d'équipements POS tiers, notamment différents scanners (**adaptabilité**)
- Gestion des paiements par carte de crédit, carte de débit ou chèque (**adaptabilité**)

PROBLÈME : FIABILITÉ – REPRISE SUR DÉFAILLANCE

- Tolérance à la panne de services distants
- Plusieurs scénarios de panne

Created by Bharat
from the Noun Project



RECOUVREMENT - MOTIVATION



Created by Bharat
from the Noun Project

- Les propriétaires des magasin ne veulent pas manquer des ventes en cas de panne
- Le développeur NextGen POS sait que les concurrents n'offrent pas ce degré de fiabilité

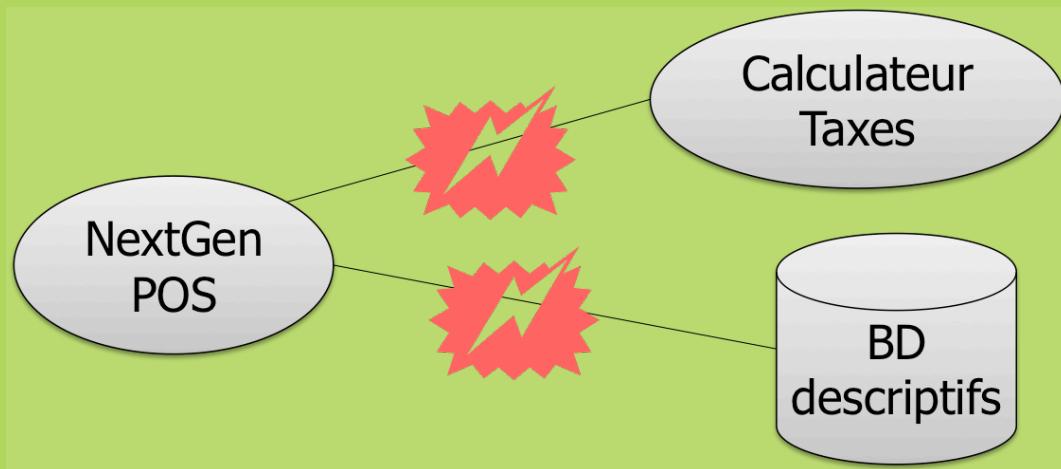


REPRISE SUR DÉFAILLANCE



Created by Bharat
from the Noun Project

- Récupération robuste en cas de défaillance
 - d'un service distant (calculateur de taxes, gestion des stocks...)
 - de la BD des produits (descriptifs et prix)



PRINCIPE DE RECOUVREMENT



Exemple: Pneu de secours permet de continuer, après un délai, perte de performance (pneu plus petit)

Created by Bharat
from the Noun Project



CONSEILS DE L'ARCHITECTE : CACHE LOCAL



Created by Bharat
from the Noun Project

- Augmente performance (et permet récupération quand l'accès à BD échoue)
- Utilise un cache local (simple fichier sur disque) des objets DescriptionProduit
- Conséquence : toujours consulter le cache local pour chercher les informations avant d'accéder au service distant
- Principe informatique « Cache »
 - [http://en.wikipedia.org/wiki/Cache_\(computing\)](http://en.wikipedia.org/wiki/Cache_(computing))
 - http://fr.wikipedia.org/wiki/Mémoire_cache

RECOUVREMENT - PRODUCT SPECIFICATION



Created by Bharat
from the Noun Project

- Recherche d'une spécification d'article
 - informations locales
 - performance
 - recouvrement
 - puis requête au système distant

RECOUVREMENT - PRODUCT SPECIFICATION



Created by Bharat
from the Noun Project

- Plusieurs niveaux de caches
 - mémoire vive (~1000)
 - performance
 - disque dur (~100 MB)
 - pannes de courant
 - serveur local au commerce
 - indépendance au réseau
 - web
 - etc.



Created by Jonathan Li
from the Noun Project

RAPPEL - ADAPTATEURS

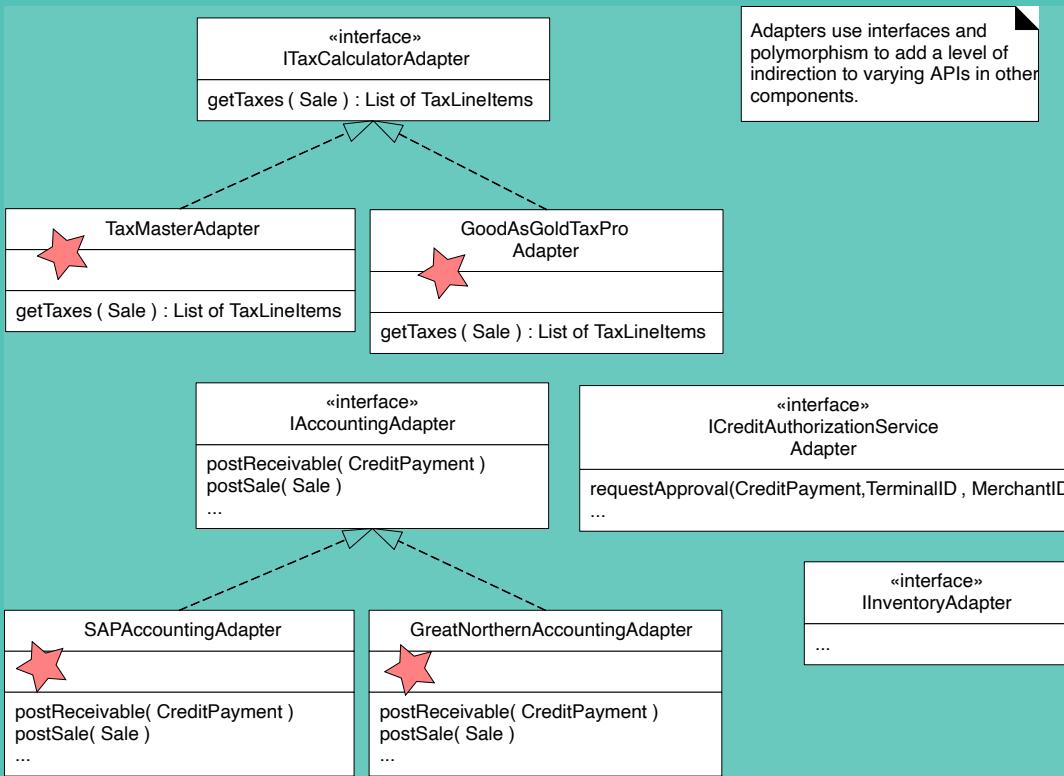
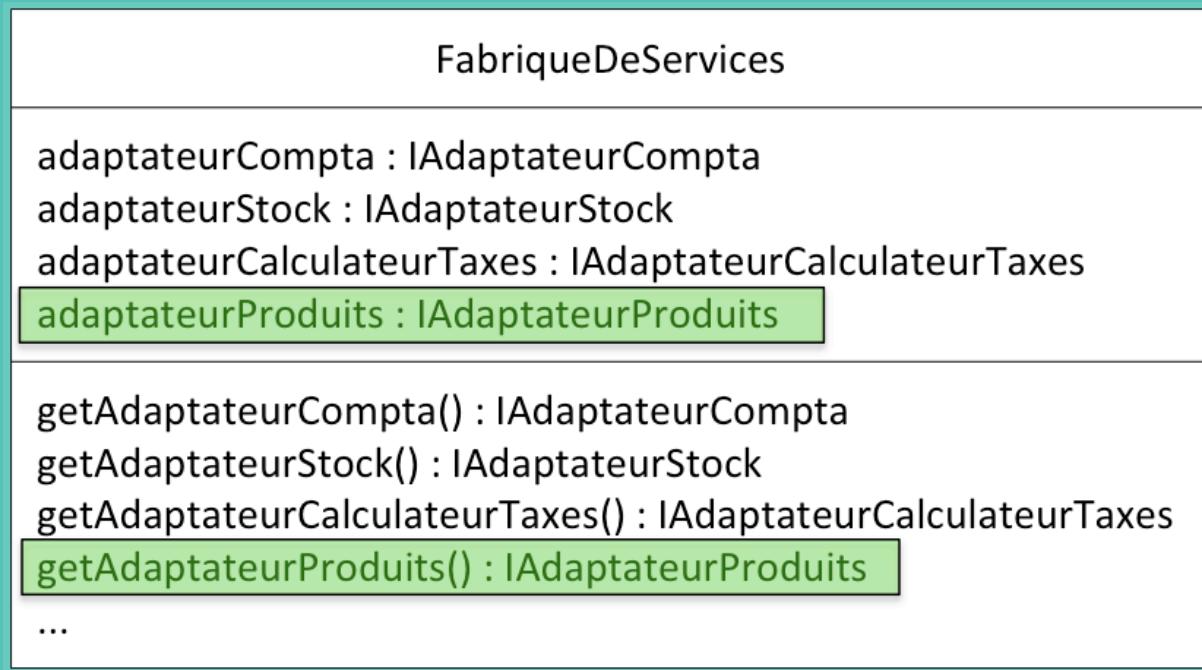


fig. F23.1

RAPPEL - ADAPTATEURS



- FabriqueDeServices retourne un adaptateur pour accéder aux variantes de services BD Produits

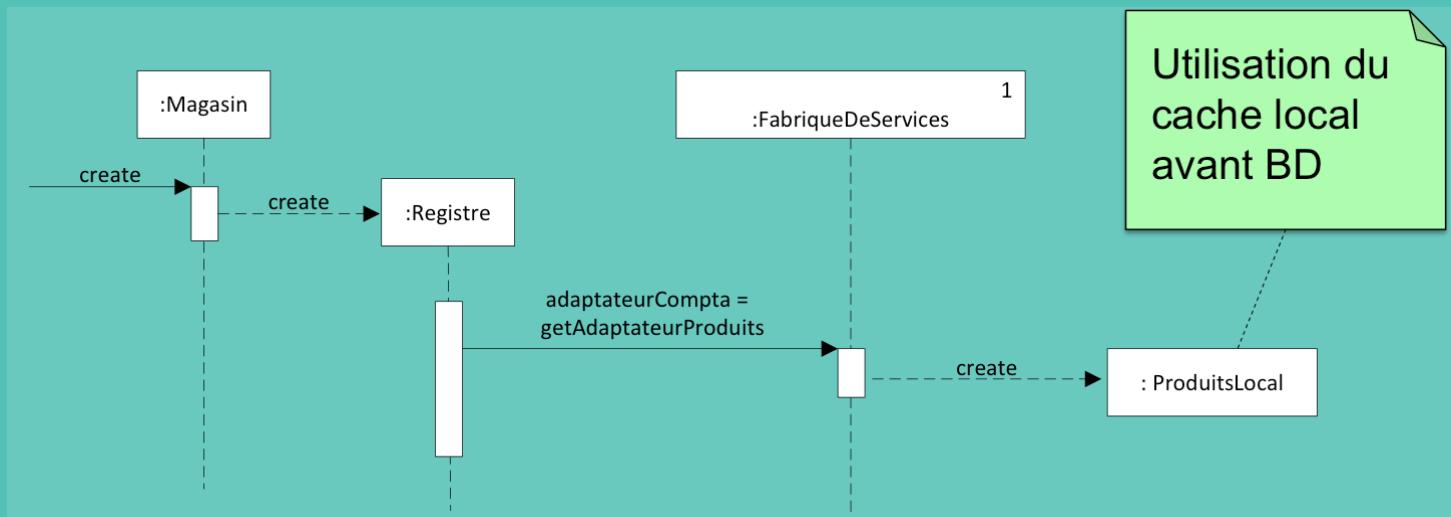


RAPPEL – ADAPTATEURS

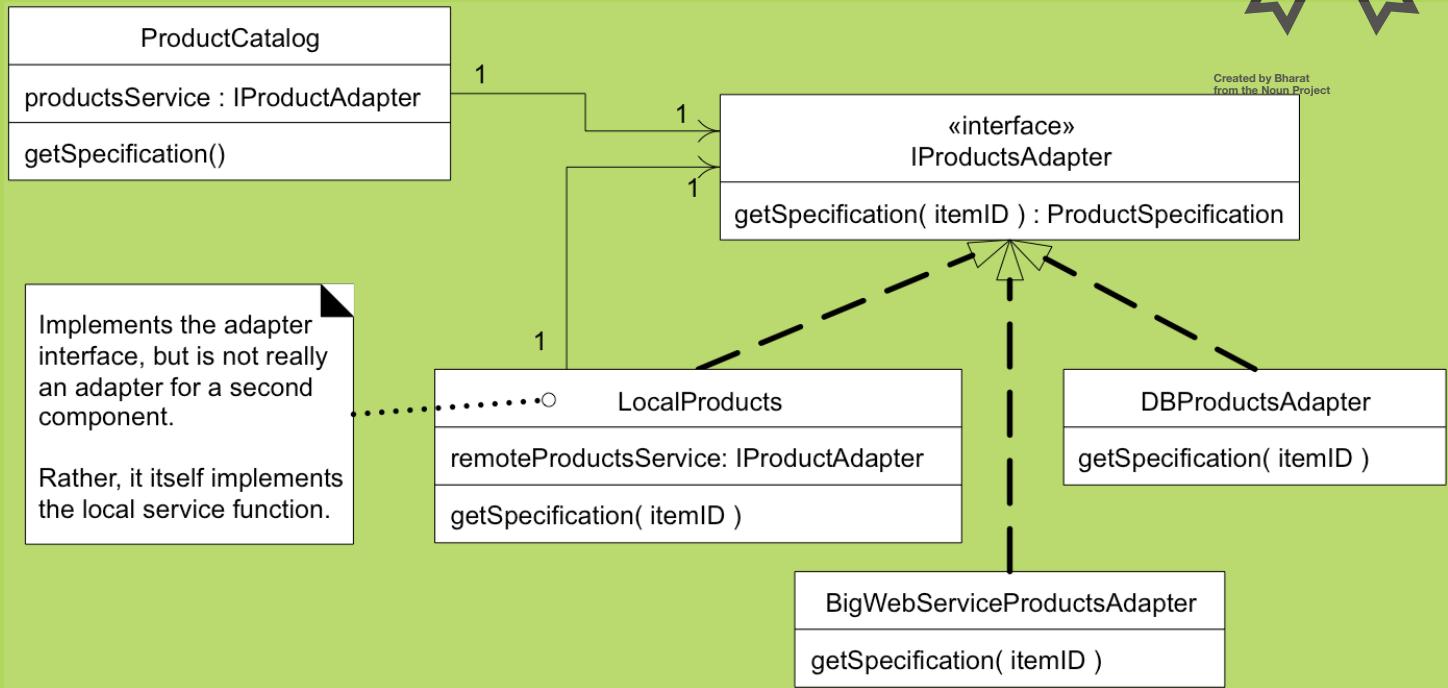


Created by Jonathan Li
from the Noun Project

- FabriqueDeService retourne un adaptateur pour accéder à la BD Produits



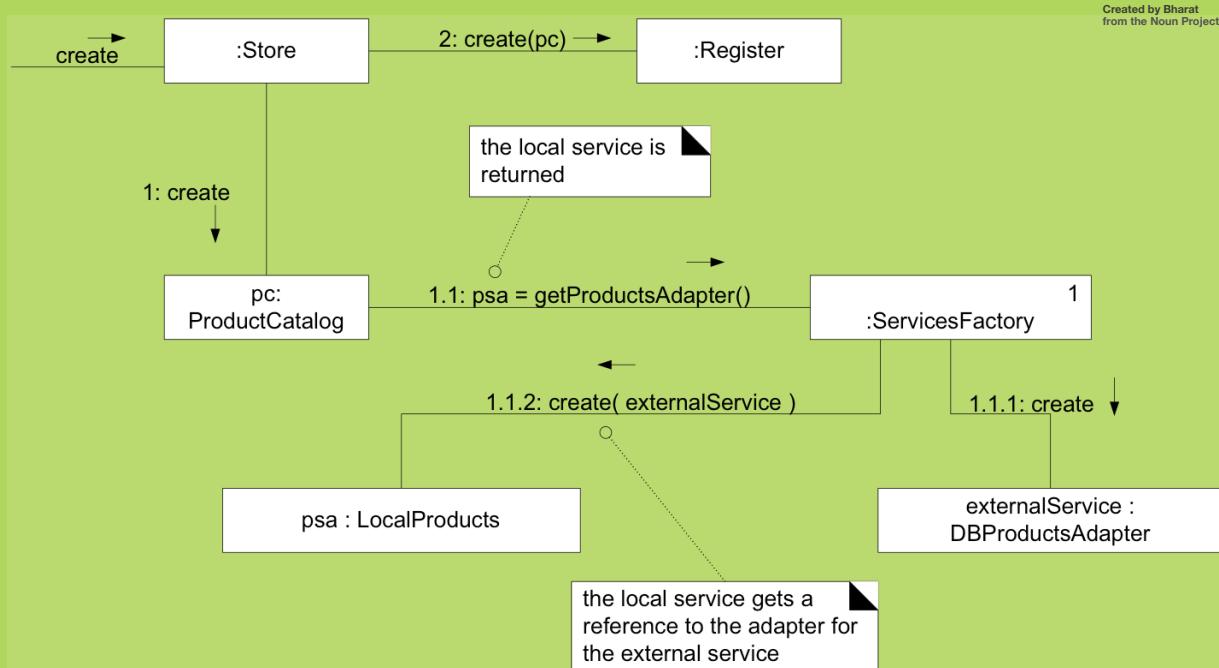
DESIGN DE RECOUVREMENT



DESIGN DE RECOUVREMENT



Initialisation



DESIGN DE RECOUVREMENT



enterItem

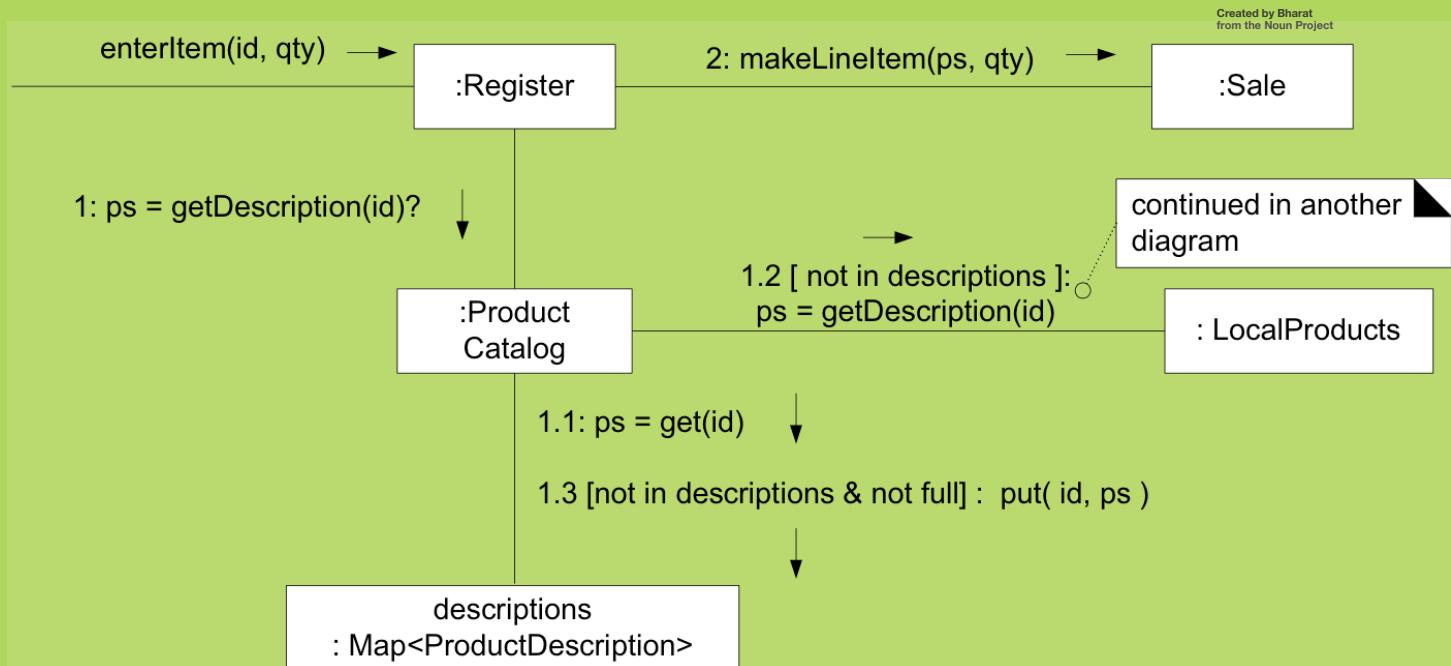


fig. F30.3

DESIGN DE RECOUVREMENT



enterItem...

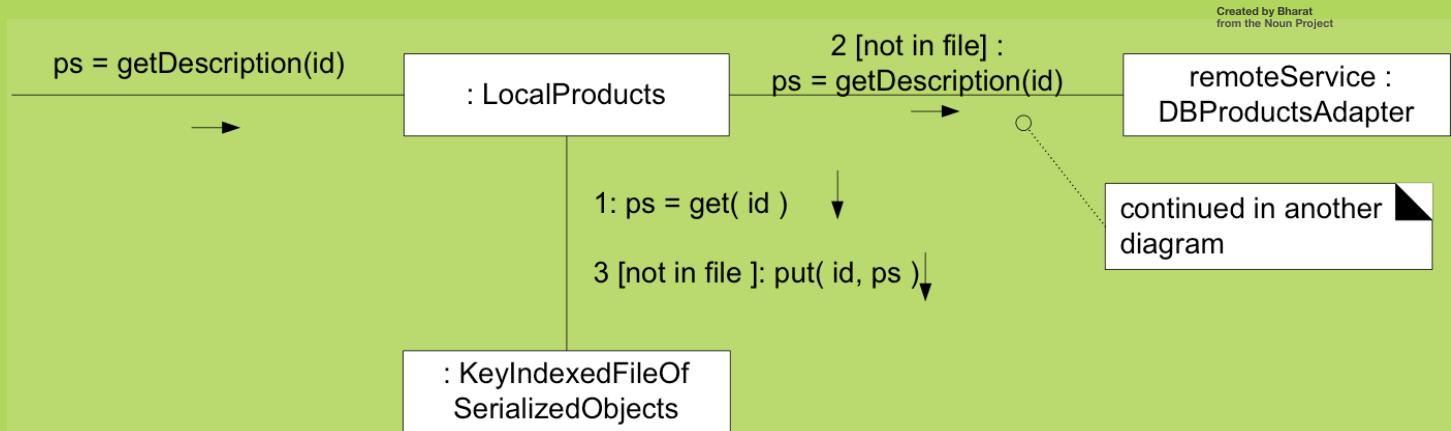
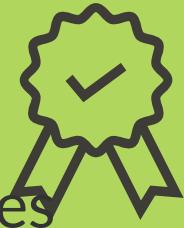


fig. F30.4

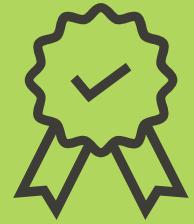
MISE EN MÉMOIRE CACHE



Created by Bharat
from the Noun Project

- Approches pour initialiser les caches
 - Paresseuse
 - au fur et à mesure
 - Stricte
 - à l'initialisation

COHÉRENCE DE CACHE



Created by Bharat
from the Noun Project

- Les prix peuvent changer
- Incohérence (niveaux de cache ↔ serveur externe)
- Mise-à-jour des caches
 - initiée par le serveur
 - initiée par le client

ATTENTION À LA « PATTERNITE »



« Patternite » : tendance de rentrer de force dans les patterns GoF

« Ça fait un boute que je n'ai pas utilisé Décorateur... »

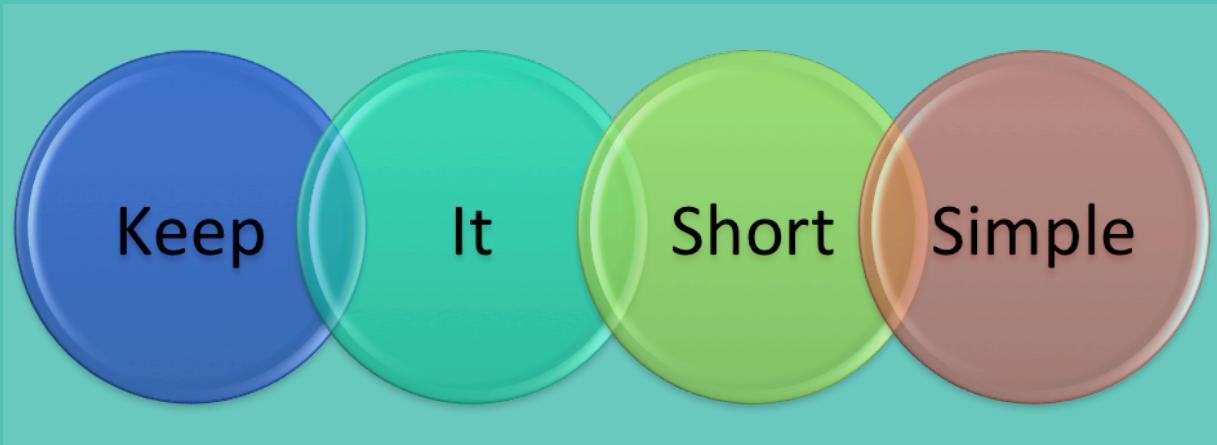


ATTENTION À LA « PATTERNITE »



Created by Jonathan Li
from the Noun Project

- Tout pattern GoF amène une complexité
- Est-ce que les bénéfices du pattern justifient la complexité?
- La simplicité (KISS) est aussi une forme d'élégance.



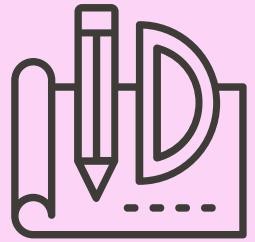
21 . 82

ATTENTION À LA « PATTERNITE »

Connaître vos patrons de conception

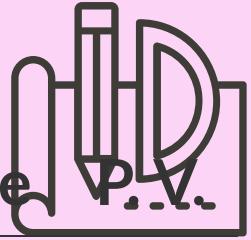
GRASP DANS LES GOF

Rapport technique



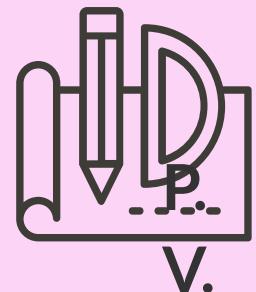
21 . 84

TABLE 7: GRASP DANS LES GOF



| Pattern GoF | Indir. | Poly. | Fab. Pure | P... P... |
|------------------|--------|-------|-----------|--------------|
| Abstract factory | ✓ | ✓ | ✓ | ✓ |
| Builder | ✓ | ✓ | ✓ | ✓ |
| Factory method | ✓ | ✓ | | ✓ |
| Prototype | ✓ | ✓ | | ✓ |
| Singleton | | | | |
| Adapter | ✓ | ✓ | ✓ | ✓ |
| Bridge | ✓ | ✓ | ✓ | ✓ |

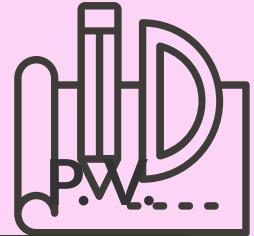
TABLE 7: GRASP DANS LES GOF



| Pattern GoF | Indir. | Poly. | Fab. Pure | V. |
|----------------------------|--------|-------|--------------|----|
| Composite | ✓ | ✓ | | ✓ |
| Decorator | ✓ | ✓ | | ✓ |
| Facade | ✓ | | ✓ | ✓ |
| Flyweight | ✓ | ✓ | ✓ | ✓ |
| Proxy | ✓ | ✓ | ✓ | ✓ |
| Chain of responsibility | ✓ | ✓ | ✓ | ✓ |

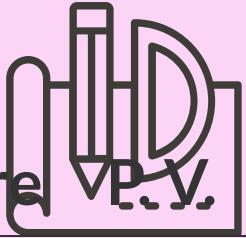
21 . 86

TABLE 7: GRASP DANS LES GOF



| Pattern GoF | Indir. | Poly. | Fab. Pure | P.W. |
|-------------|--------|-------|-----------|-----------|
| Command | ✓ | ✓ | ✓ | ✓ |
| Interpreter | ✓ | ✓ | ✓ | ✓ |
| Iterator | ✓ | ✓ | ✓ | ✓ |
| Mediator | ✓ | ✓ | ✓ | ✓ |
| Memento | ✓ | | ✓ | ✓ |
| Observer | ✓ | ✓ | ✓ | ✓ |
| State | ✓ | ✓ | | ✓ |

TABLE 7: GRASP DANS LES GOF



| Pattern GoF | Indir. | Poly. | Fab. | Pure |
|-----------------|--------|-------|------|------|
| Strategy | ✓ | ✓ | | ✓ |
| Template method | ✓ | ✓ | | ✓ |
| Visitor | ✓ | ✓ | ✓ | ✓ |

IDENTIFICATION DES GRASP DANS LES LOGOS

<https://docs.google.com/document/d/1Mmi0SDH4Ex4Uusp=sharing>

SOLUTION

<https://docs.google.com/document/d/1pe3RTLkBFIHErDrne9et8qqyTinNfcA21HIT-o/edit?usp=sharing>

MODULE DIAGRAMMED'ÉTAT

1. Diagramme d'état - 26.34m
2. Diagrammes d'état - exercice - 02.54m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

DIAGRAMME D'ÉTAT



Pour modéliser les comportements.

Created by Rudez Studio
from the Noun Project

Notion préalable: Automate fini

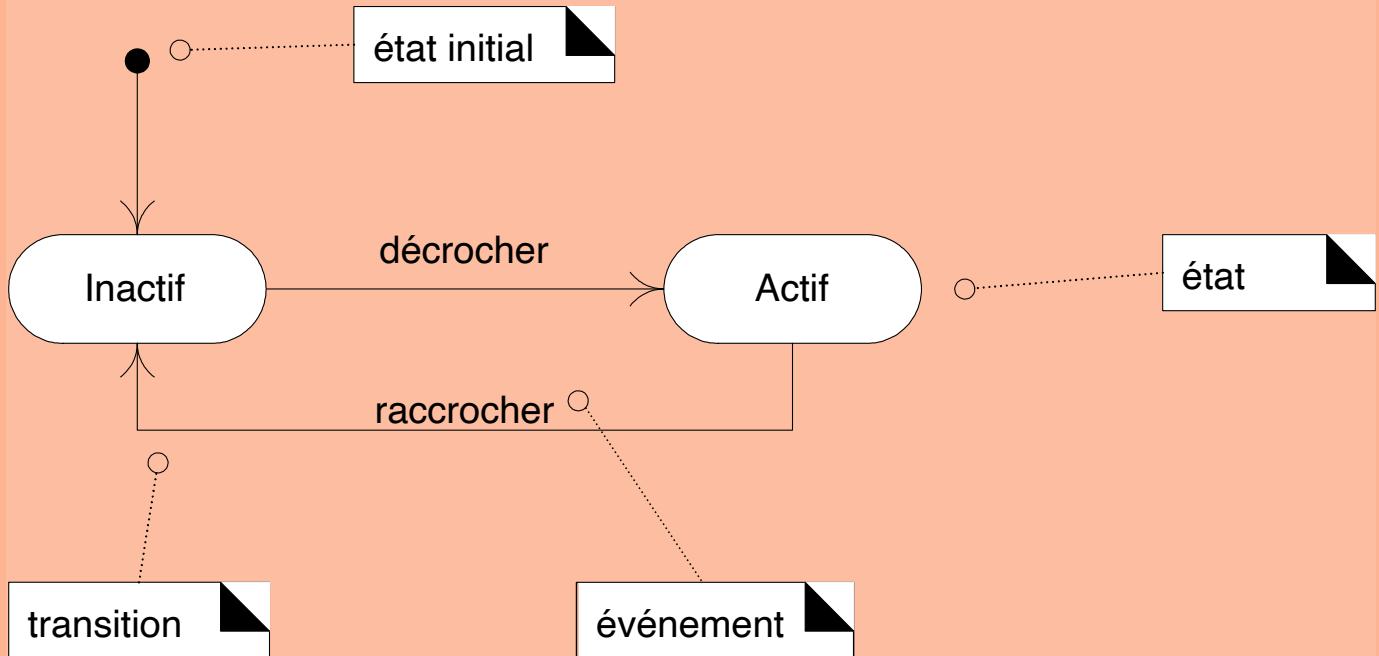
22 . 2

EXEMPLE D'AUTOMATE FINI

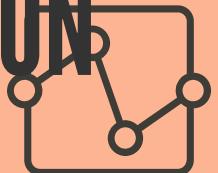


Téléphone

Created by Rudez Studio
from the Noun Project



ÉVÉNEMENT, ÉTAT ET TRANSITION



Created by Rudez Studio
from the Noun Project

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

OBJETS ÉTAT-DÉPENDANTS ET ÉTAT-INDÉPENDANTS



Created by Rudez Studio
from the Noun Project

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

MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS

- Équipement physiques contrôlés par des logiciels

Created by Rudez Studio
from the Noun Project



MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS



Transactions et objets métier apparentés

Created by Rudez Studio
from the Noun Project

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



22 . 7

MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS



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

Created by Rudez Studio
from the Noun Project

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



MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS



Created by Matheus Studio
from the Noun Project

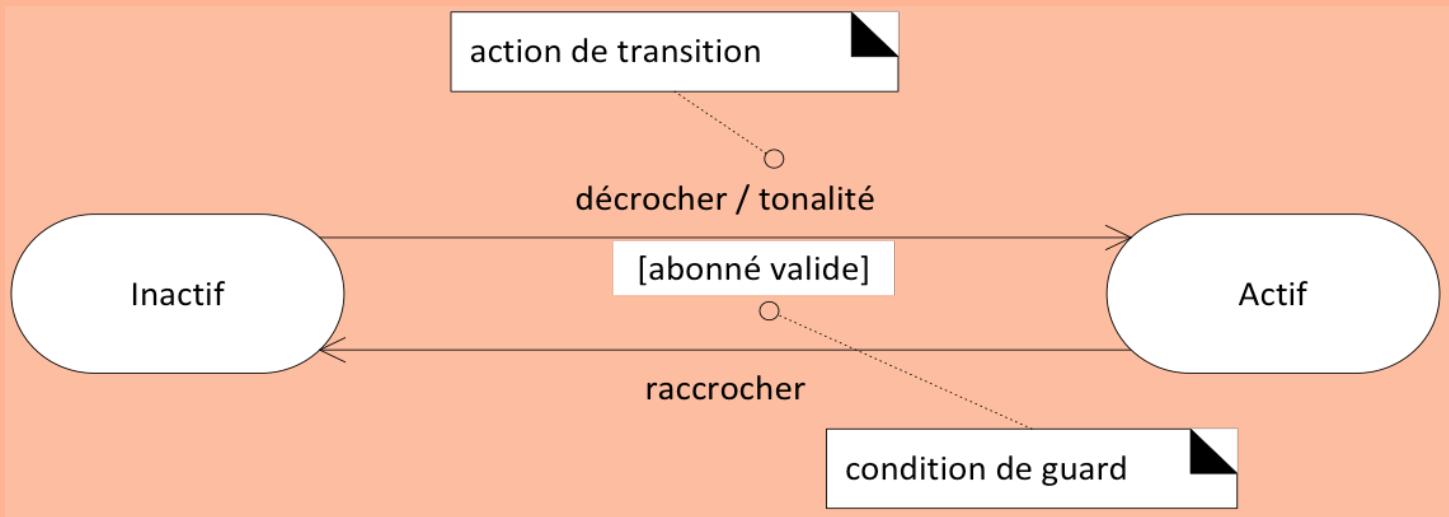
- Protocoles de communication
 - p.ex. en TCP, si le gestionnaire du protocole est dans l'état « fermé », un message « close » sera ignoré
- Flot page/fenêtre IHM
 - séquence légale des pages Web ou des fenêtres, souvent complexe
- Contrôleurs de flot ou de sessions
 - objets « session » p.ex. application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

NOTATION ADDITIONNELLE



Created by Rudez Studio
from the Noun Project

Transitions peuvent avoir des actions et des conditions de guard

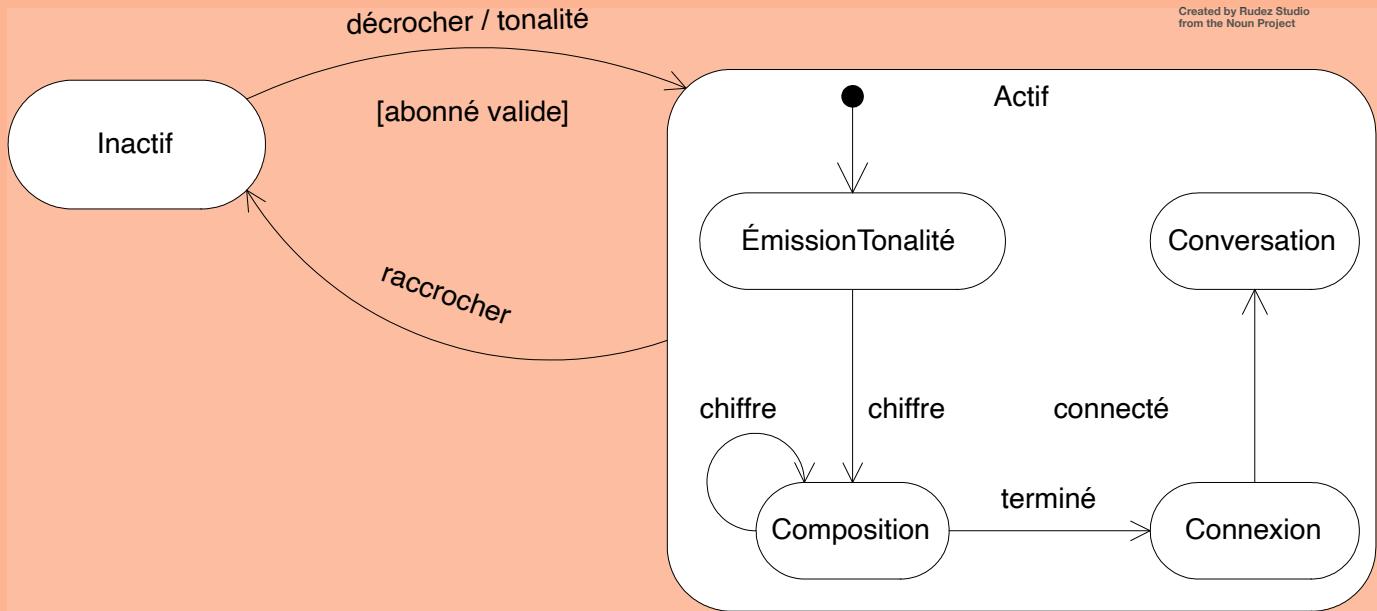


NOTATION ADDITIONNELLE

États imbriqués

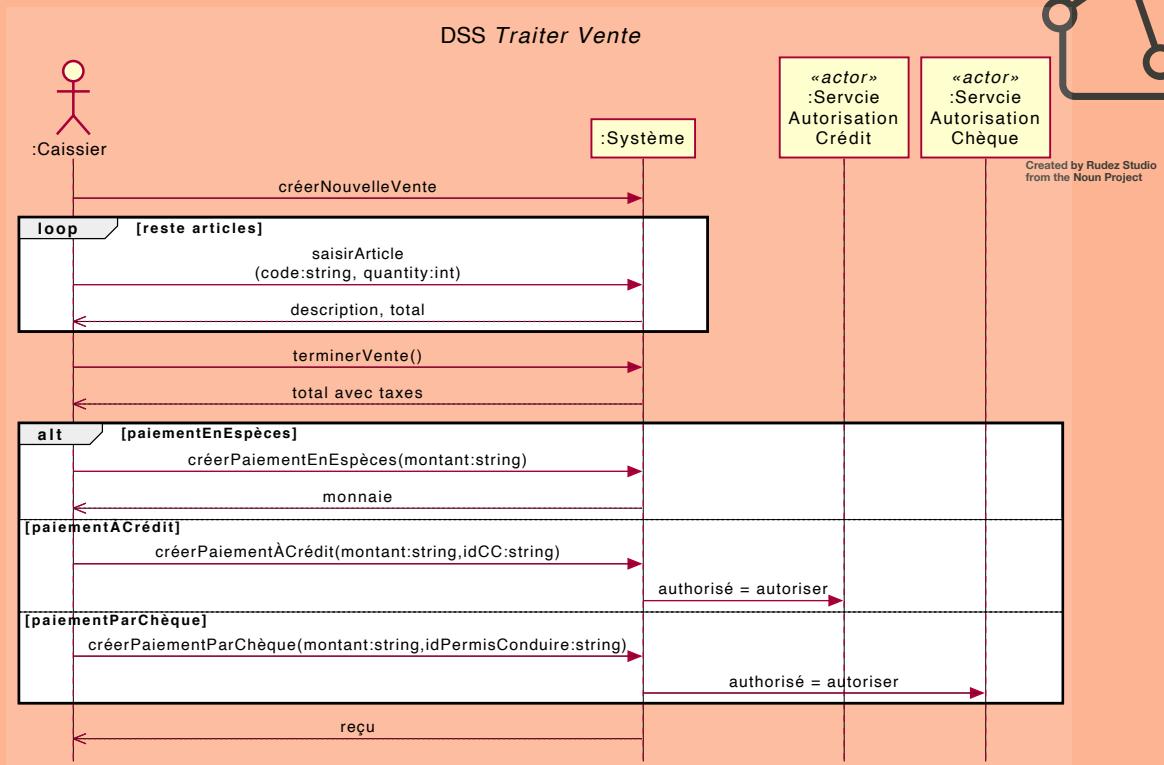


Created by Rudez Studio
from the Noun Project



- Pouvez vous réaliser le diagramme de classe correspondant?

*AUTOMATE FINI ET CAS D'UTILISATION 1/2

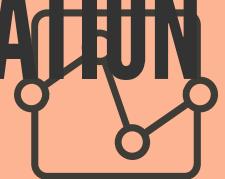


22 . 12

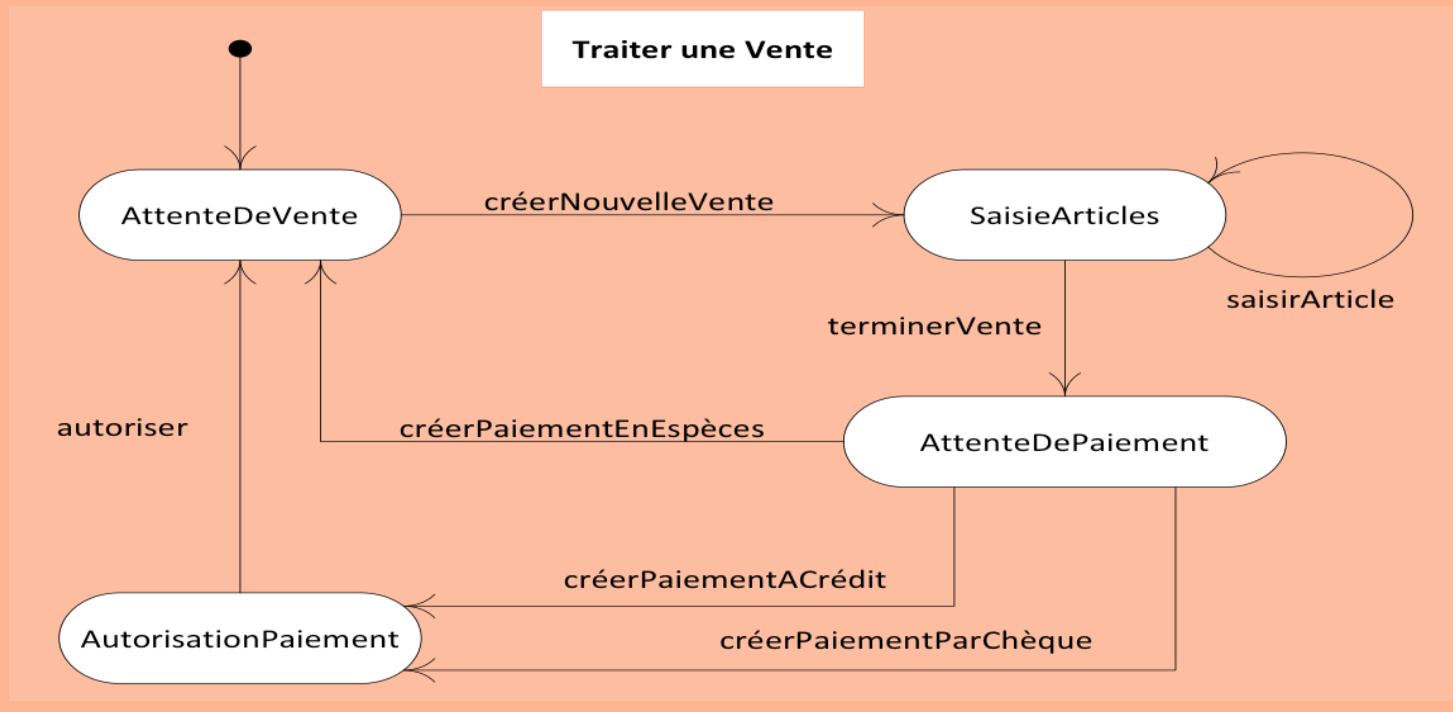
- Diagramme d'état correspondant?

*AUTOMATE FINI ET CAS D'UTILISATION

2/2

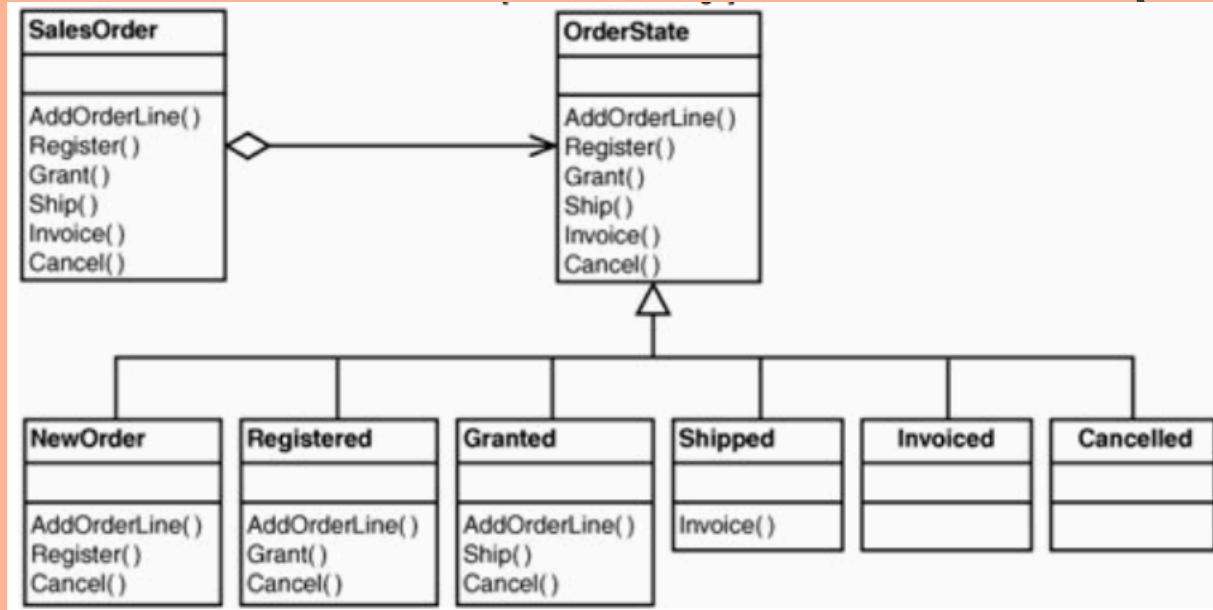
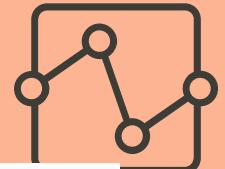


Created by Rudez Studio
from the Noun Project



22 . 13

ÉTAT D'UNE COMMANDE



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

EXERCICES DIAGRAMME D'ÉTAT



Created by Rudez Studio
from the Noun Project

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

DIAGRAMME D'ÉTAT IMPLÉMENTATION



Comment implémenter?

Created by Rudez Studio
from the Noun Project

1. StackOverflow: Is there a typical state machine implementation pattern?
2. Derek Banas, State pattern

DIAGRAMME D'ÉTAT

EXERCICE VIDÉO PROJECTEUR

- En équipe de 4, réaliser le diagramme d'état en suivant les étapes identifiées dans l'exercice suivant:

<https://github.com/yvanross/LOG210-exercices/blob/master/etat/etat-videoprojecteur.md>

MODULE DIAGRAMME D'ACTIVITÉ

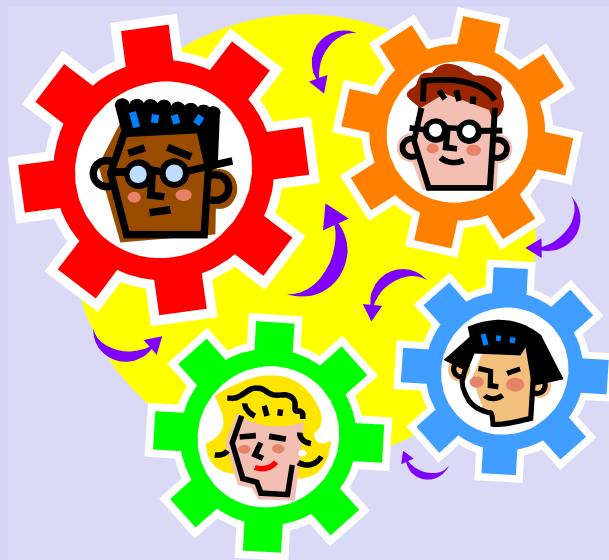
1. Diagramme d'activité - 16.01m
2. Diagrammes d'activité - exercice - 01.18m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

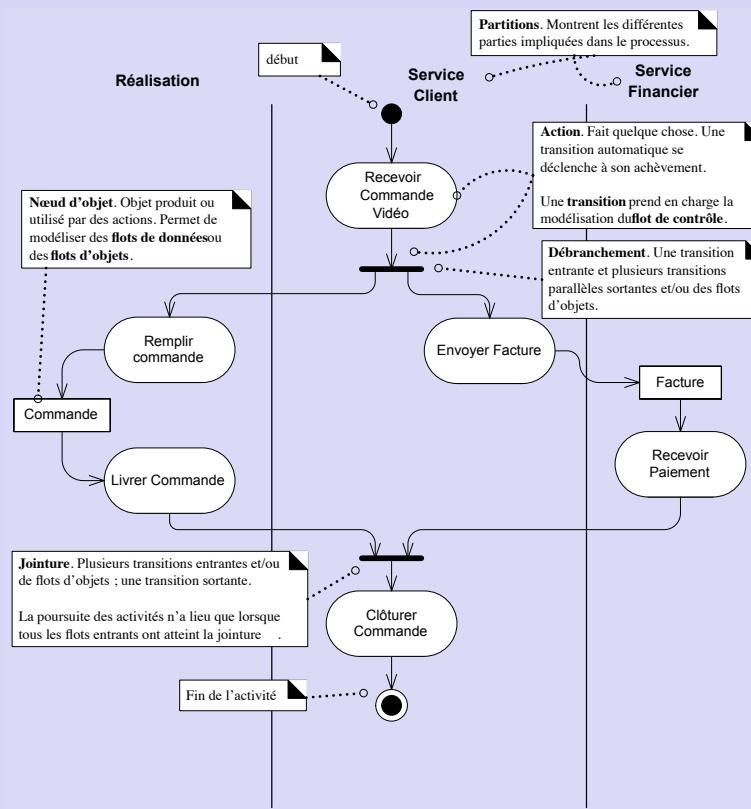
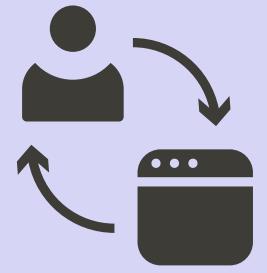
DIAGRAMMES D'ACTIVITÉS



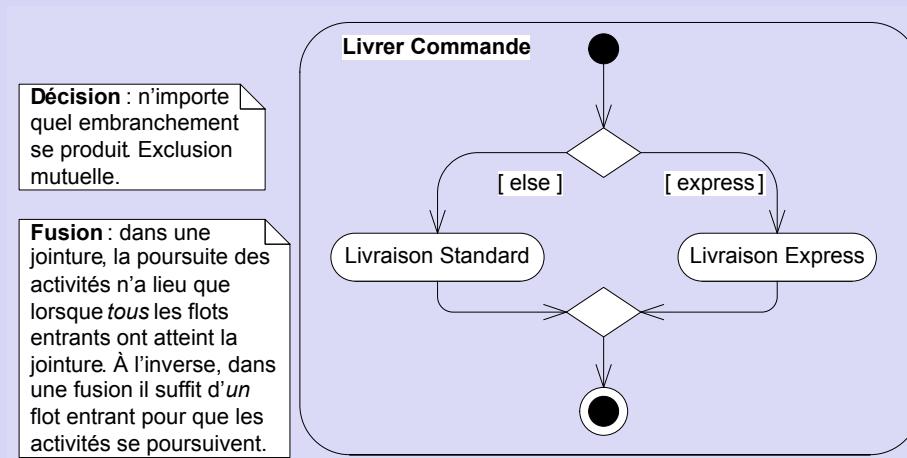
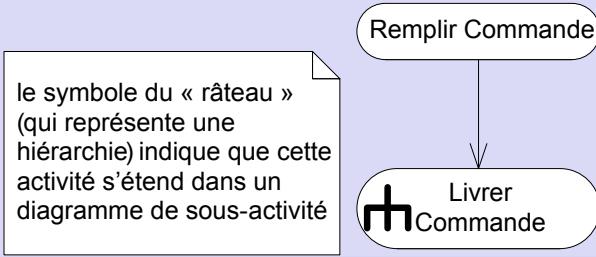
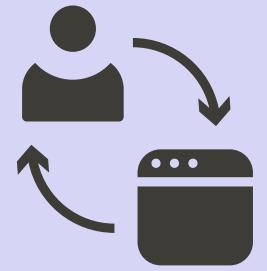
- Exposer les activités séquentielles et parallèles d'un processus.



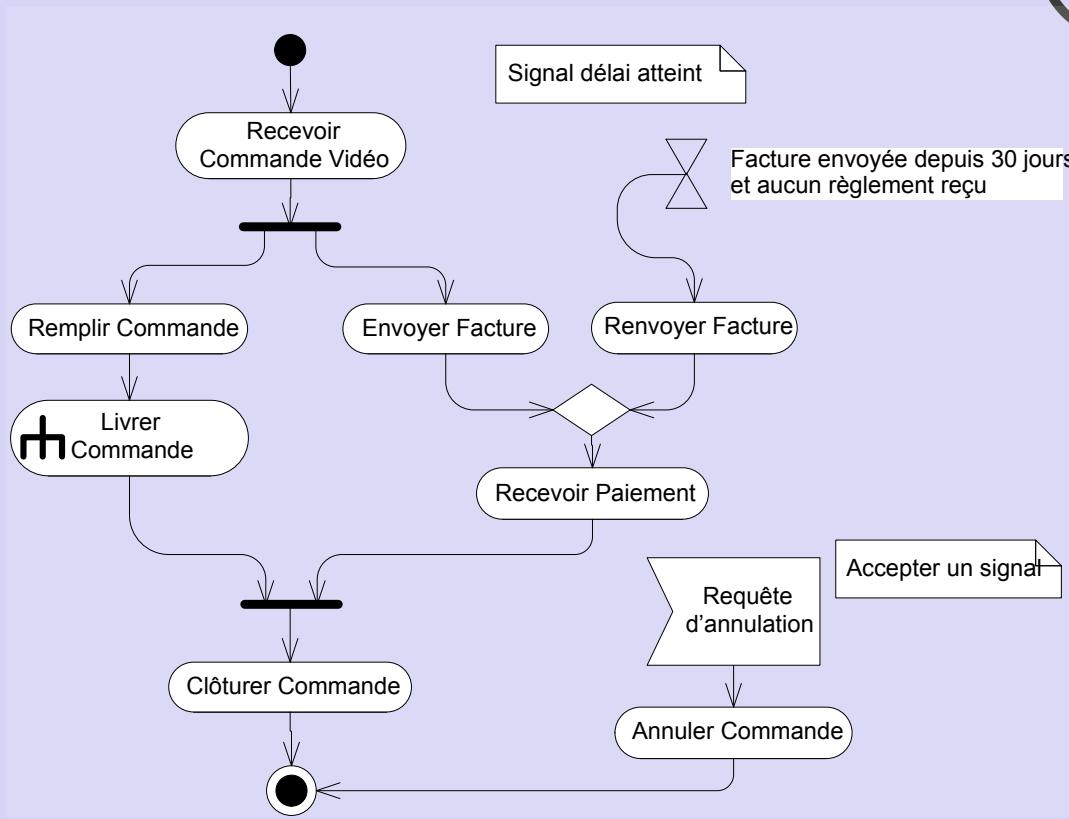
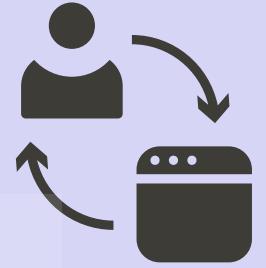
DIAGRAMMES D'ACTIVITÉS



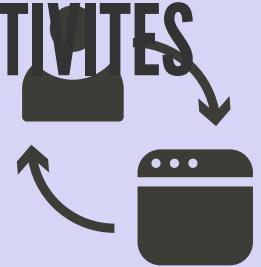
COMPLÉMENTS SUR LA NOTATION



SIGNALS

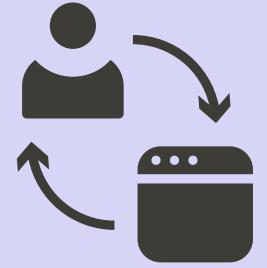
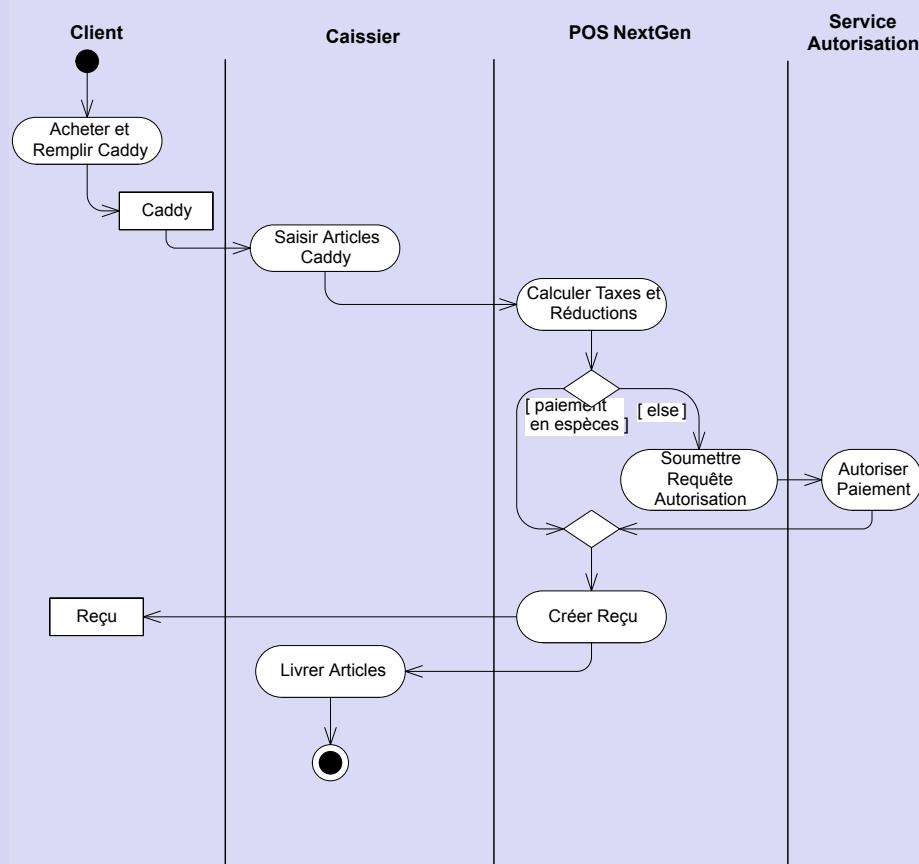


LIGNES DIRECTRICES – MODÉLISATION D'ACTIVITÉS

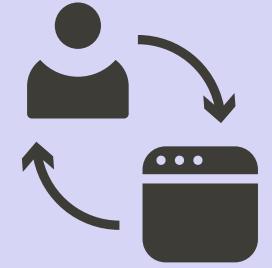
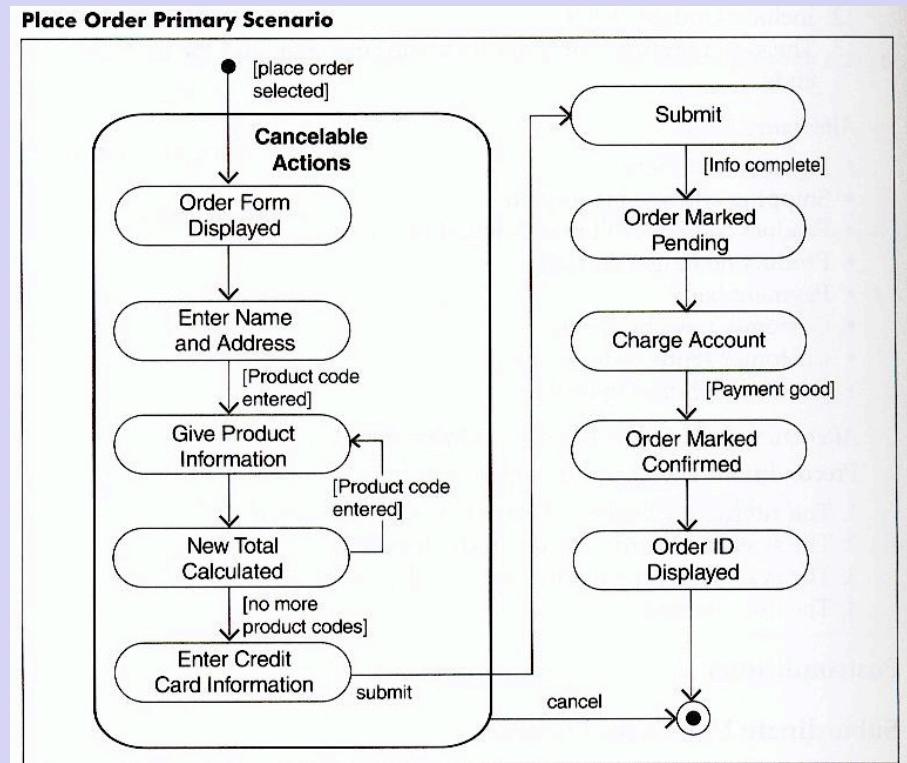


- Utile pour les processus très complexes
 - impliquant plusieurs parties
 - impliquant des activités séquentielles
- ■ impliquant des processus en parallèles
- Utilisez les niveaux d'abstraction («rateau» et sous-activité) afin de gérer la complexité
 - Niveaux 0, 1, 2, etc.
 - Dans un diagramme, veillez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

MODÉLISATION DE TRAITER UNE VENTE

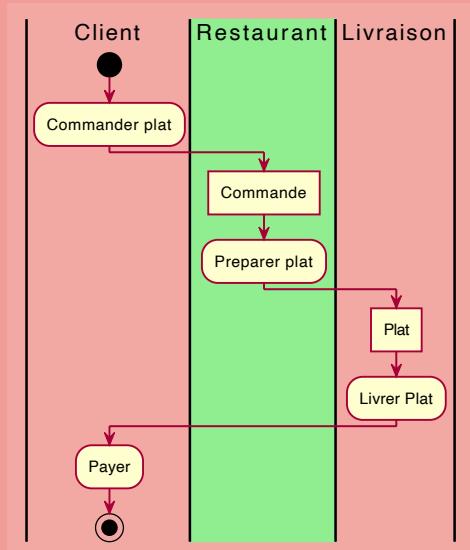


D'AUTRES EXEMPLES



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

OUTILS PLANTUML



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

Exemple : Dynamique GitHub Classroom

EXERCICE: RETOUR DE VOITURE LOUÉ



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

Scénario

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

*EXERCICES - MODÉLISER D'UNE DYNAMIQUE DE STATIONNEMENT

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

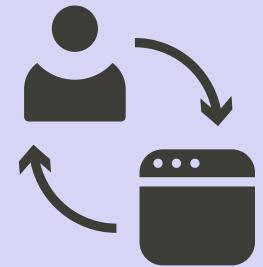
EXERCICES DIAGRAMME D'ACTIVITÉ



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

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

DIAGRAMME D'ACTIVITÉ



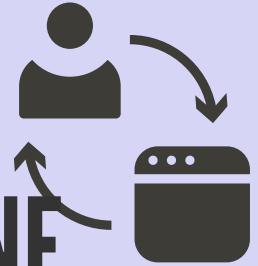
Comment implémenter?

Business Process Modeling Notation (BPMN)

1. BPMN Demo
2. UML
3. Derek Banas, Activity Diagram

DIAGRAMME D'ACTIVITÉ

EXERCICE RECETTE DE CUISINE



- En équipe de 4, réaliser le diagramme d'activité en suivant les étapes identifiées dans l'exercice suivant:

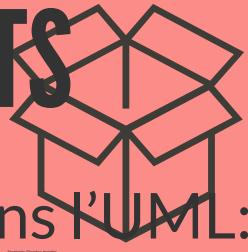
<https://github.com/yvanross/LOG210-exercices/blob/master/activite/activite-recette-de-cuisine.md>

MODULE DIAGRAMME DE DOMPOSANT ET DÉPLOIEMENT

2. Diagrammes de composants - 17.37m
3. Diagrammes de déploiement - 10.02m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

DIAGRAMMES DE COMPOSANTS



Un composant n'est pas clairement défini dans l'UML:

« Un composant est la **partie modulaire** d'un système qui encapsule son contenu et dont la manifestation est **remplaçable** dans son environnement. Il définit son **comportement** en termes d'**interfaces** fournies et requises. Ainsi, un composant peut être utilisé comme un type dont la conformité est définie par ces interfaces fournies et requises. »

EXEMPLE EN PLANTUML

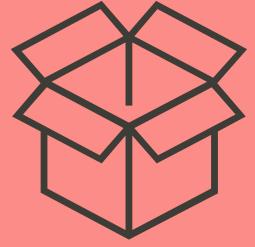
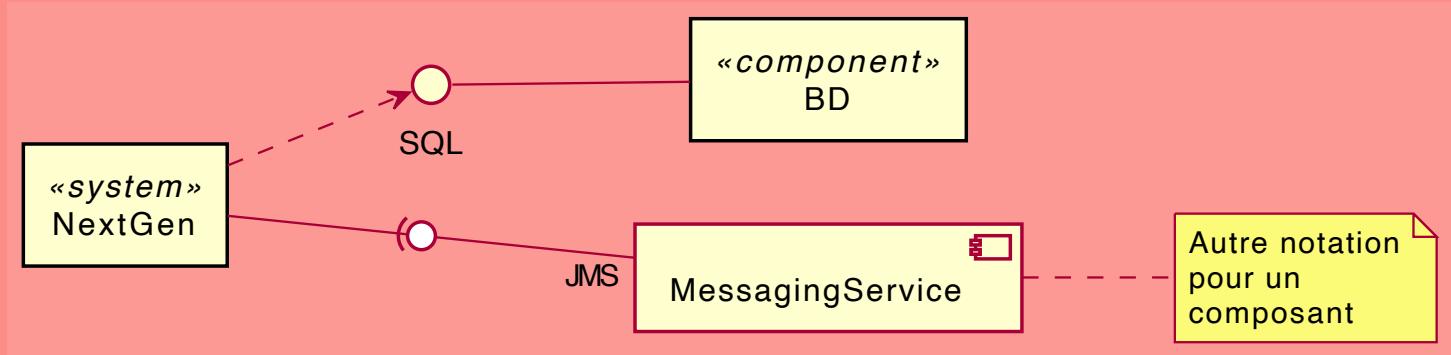


Figure F31.2 Composants UML



(PlantUML)

EXISTE-T-IL RÉELLEMENT DES COMPOSANTS LOGICIELS?

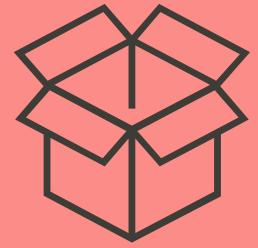


They have to exist. Sales and marketing people are talking about them.
Components are not a technology. Components are about how customers want to relate to software.

- They want to be able to buy their software a piece at a time, and to be able to upgrade it just like they can upgrade their stereo.
- They want new pieces to work seamlessly with their old pieces, and to be able to upgrade on their own schedule, not the manufacturer's schedule.
- They want to be able to mix and match pieces from various manufacturers. This is a very reasonable requirement. It is just hard to satisfy.

RalphJohnson do component exist

MÉTAPHORE CINÉMA MAISON



- 1. Heroku
- IFTTT
- 2. IOT



viet / photostream

DIAGRAMME DE COMPOSANTS

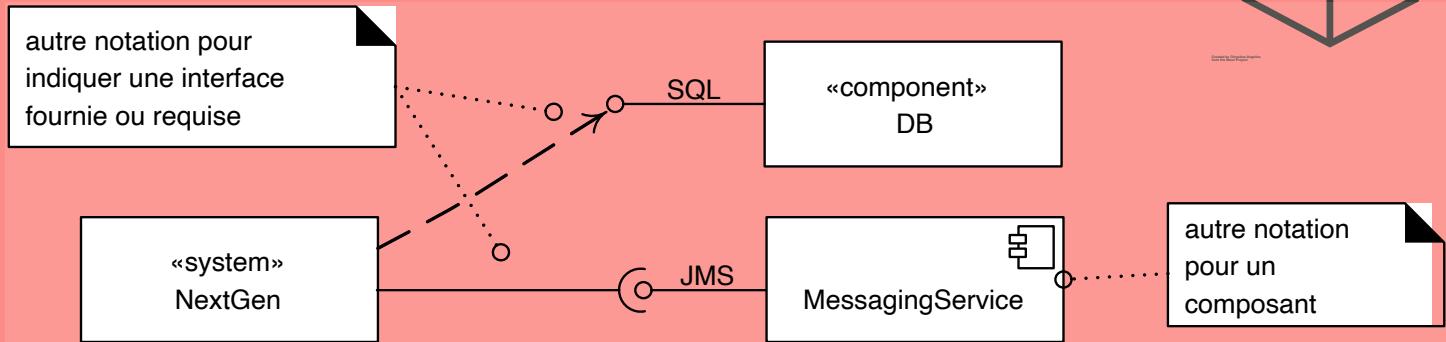


fig: 31.2, A37.2

D'AUTRES EXEMPLES

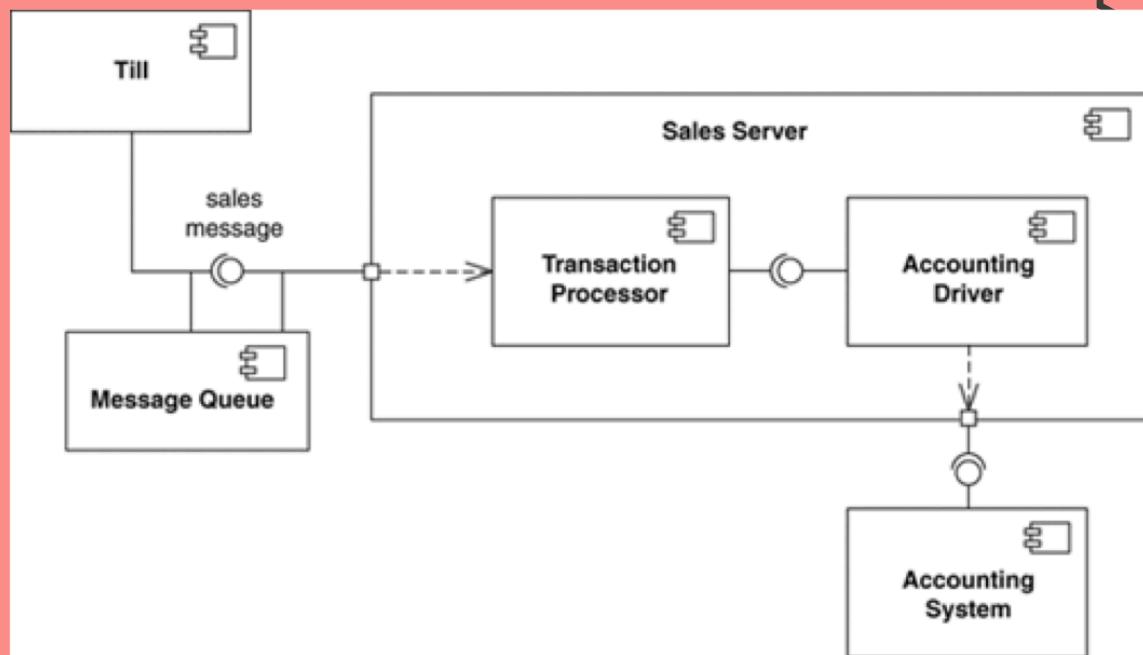
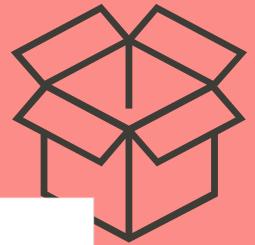
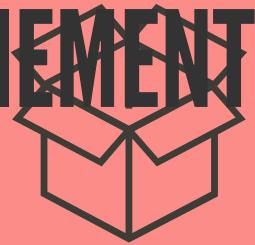


fig.: F14.2 UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition

EXERCICES DIAGRAMME DE DÉPLOIEMENT

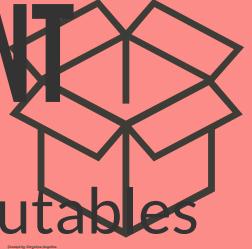


1. Application web
2. Tutorial online
3. UML deployment diagram - Apple iTunes
4. UML Deployment Diagrams Examples

Prenez le temps regarder la documentation de planUml pour les diagrammes de déploiement et de composants. Nous ferons un exercice la semaine prochaine.

- <https://plantuml.com/deployment-diagram>
- <https://plantuml.com/component-diagram>

DIAGRAMMES DE DÉPLOIEMENT

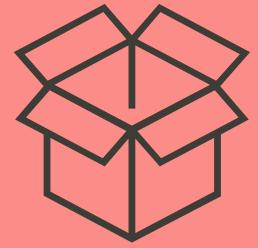


Documenter (1) comment les fichiers exécutables seront affectés sur les nœuds de traitement et (2) la communication entre composants physiques



- <http://wahnetwork.com/2012/06/03/hp-discover-2012-targeted-sessions/>
- <http://gadgetsin.com/speck-pixelskin-hd-wrap-ipad-2-case.htm>
- [http://wiki.openhr/wiki/Samsung_I7500_\(Samsung_Galaxy\)](http://wiki.openhr/wiki/Samsung_I7500_(Samsung_Galaxy))

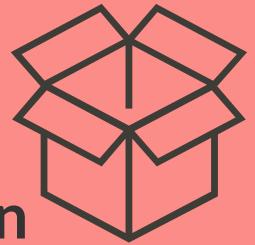
TYPES DE NŒUDS



Nœud physique (équipement)

Ressource de traitement physique (p.ex. de l'électronique numérique), dotée de services de traitement et de mémoire destinés à exécuter un logiciel. Ordinateur classique, cellulaire, etc.

TYPES DE NŒUDS



Nœud d'environnement d'exécution (EEN *execution environment node*)

Ressource de traitement logiciel qui s'exécute au sein d'un nœud externe (comme un ordinateur) et offrant lui-même un service pour héberger et exécuter d'autres logiciels.

NŒUD D'ENVIRONNEMENT EXÉCUTION



- Système d'exploitation (OS) est un logiciel qui héberge et qui exécute des programmes
- Machine virtuelle (JVM ou .NET)
- Moteur de base de données (p.ex. PostgreSQL) exécute les requêtes SQL
- Navigateur Web héberge et exécute JavaScript, applets Flash/Java
- Moteur de workflow
- Conteneur de servlets ou conteneur d'EJB

DIAGRAMME DE DÉPLOIEMENT

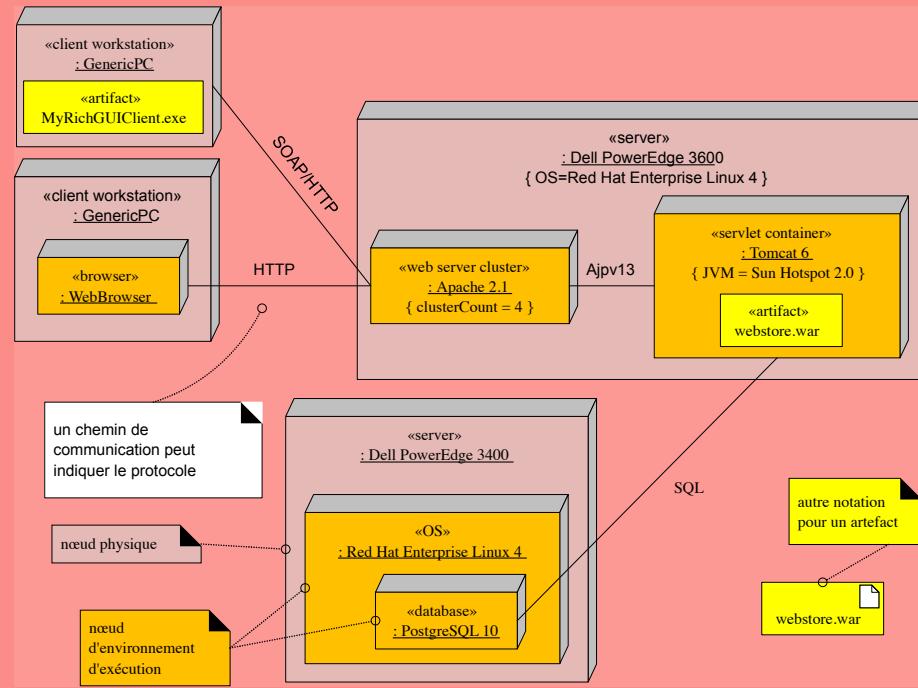
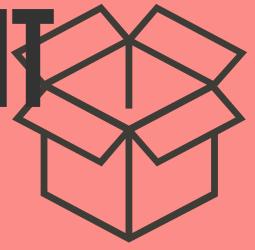


fig. F31.1, A37.1

DIAGRAMME DE DÉPLOIEMENT

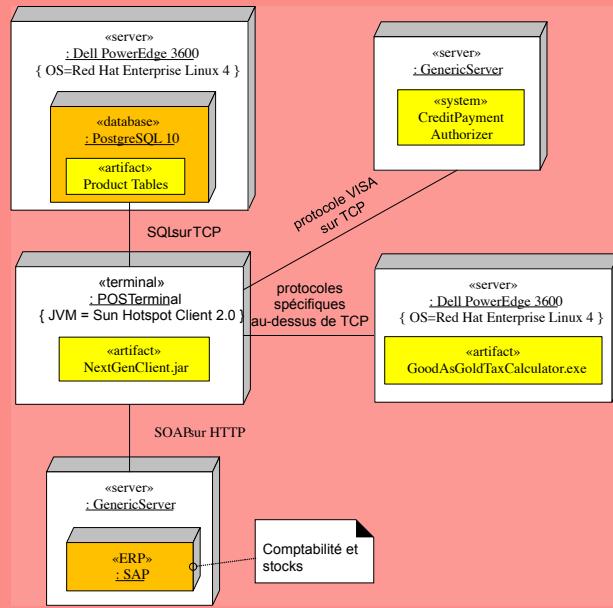
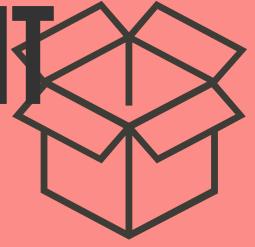
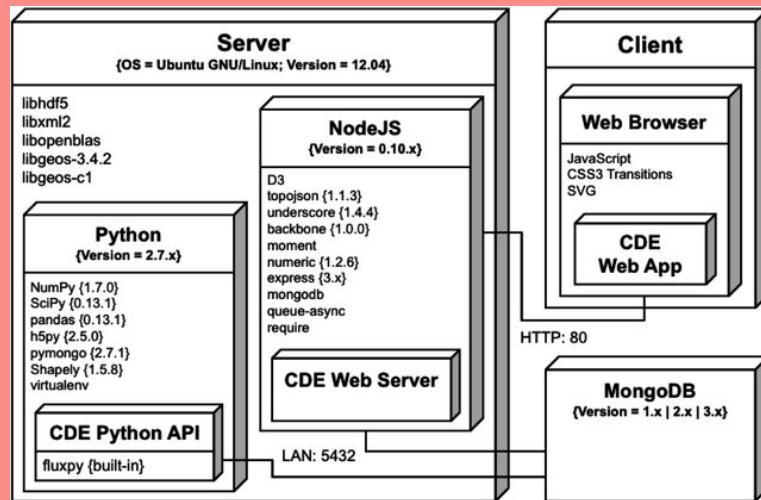


fig. F32.2, A38.2

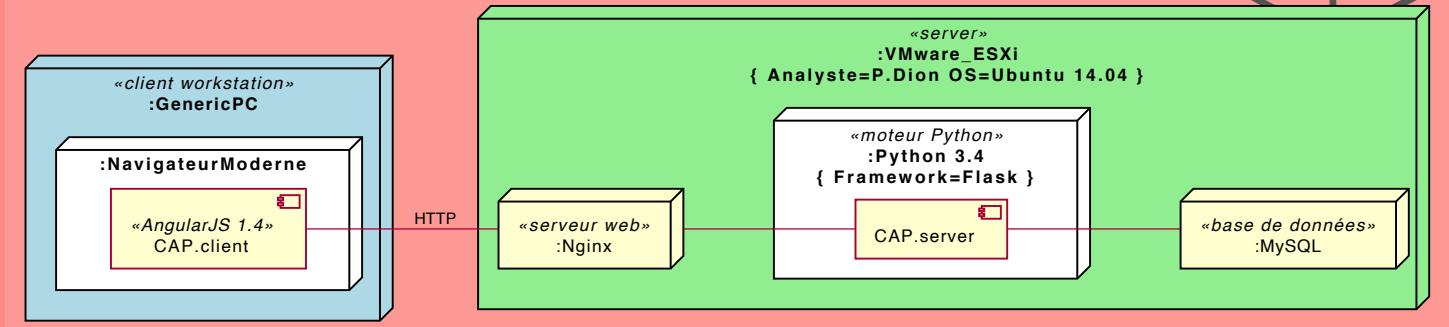
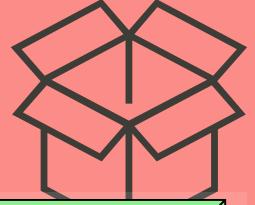
EXEMPLE 1/



La figure 2 de **Distributed visualization of gridded geophysical data: a web API for carbon flux:**

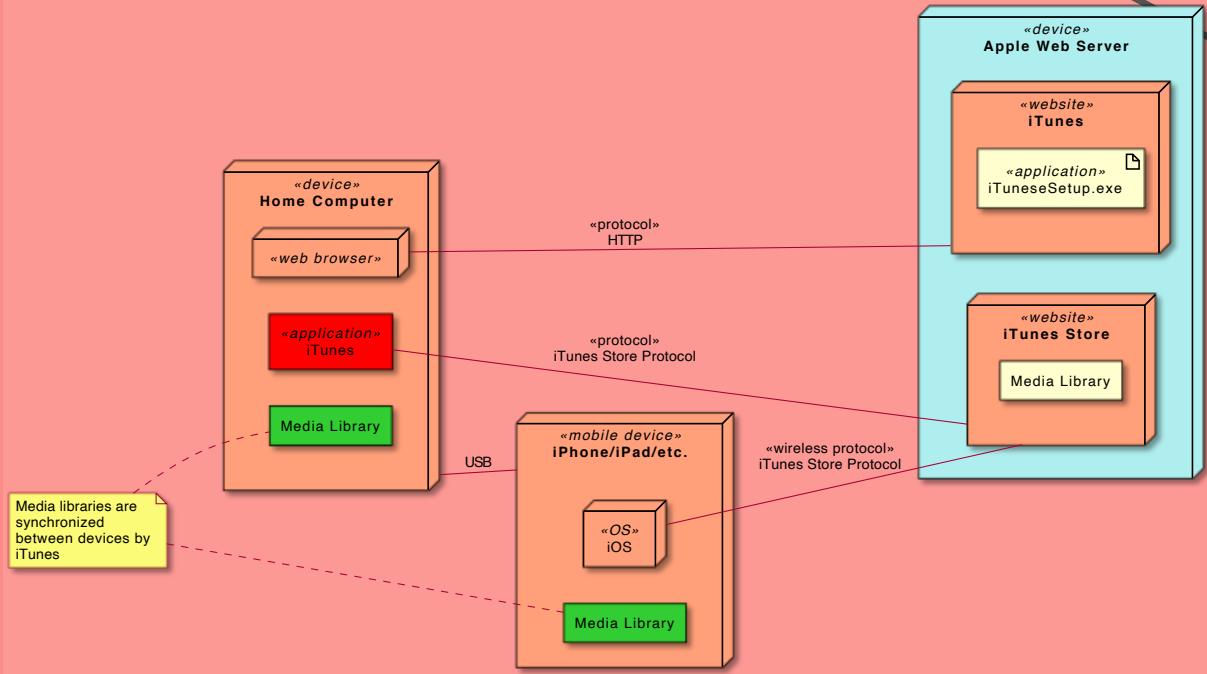
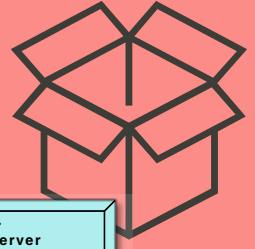


EXEMPLE 2/



(PlantUML)

EXAMPLE 3/



(PlantUML)

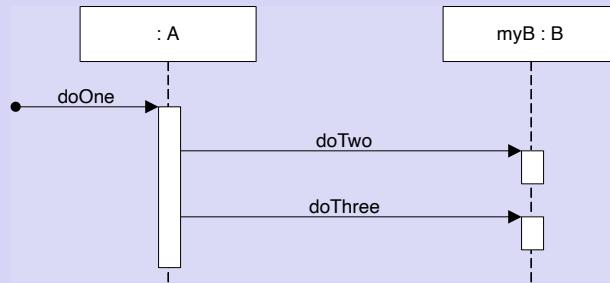
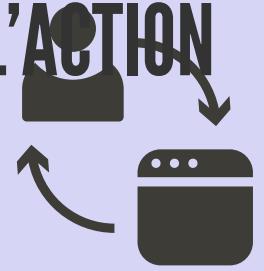
MODULE AUTRE DIAGRAMME UML

1. Diagramme d'interaction - 09.33m
2. Révision diagrammes UML - 16.57m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

DIAGRAMMES D'INTERACTION MODÉLISENT L'ACTION

Diagrammes de séquence



Diagrammes de communication

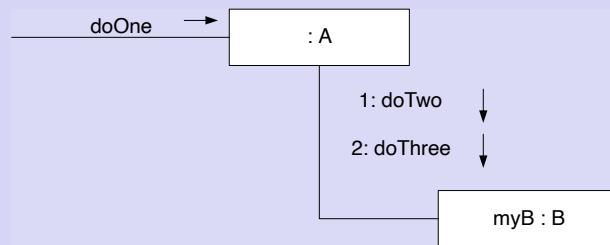
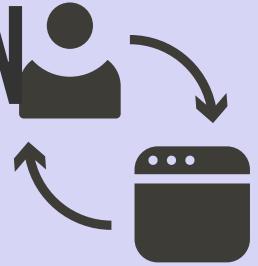


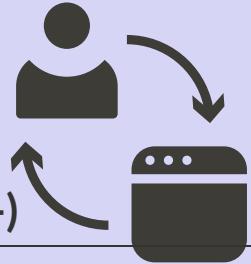
DIAGRAMME D'INTERACTION



- Pour détailler les opérations
- Annoté avec les GRASP

25 . 3

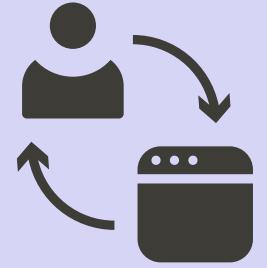
COMPARAISON



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

RAPPEL DE LOG121

CLASSE VS. INSTANCE

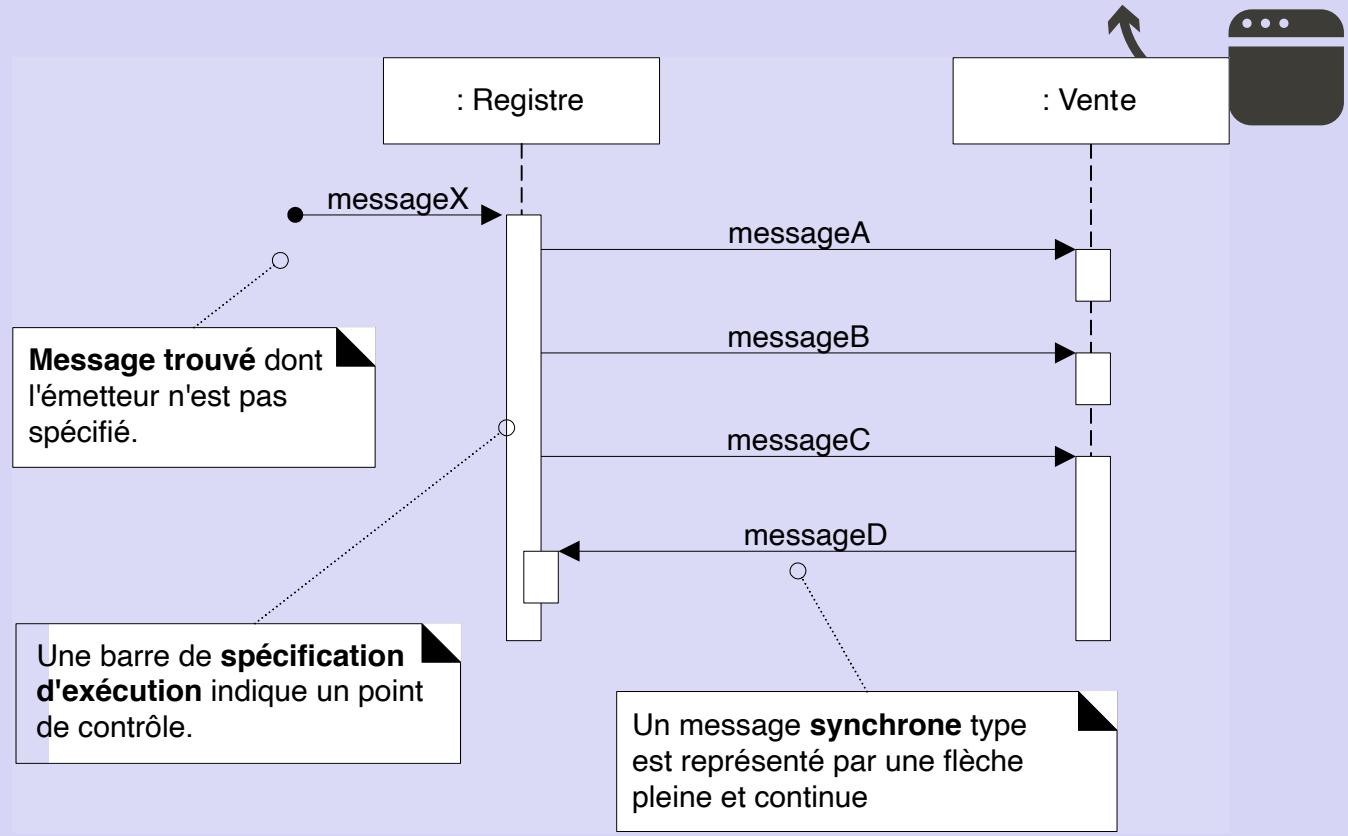


Vente - - - Classe

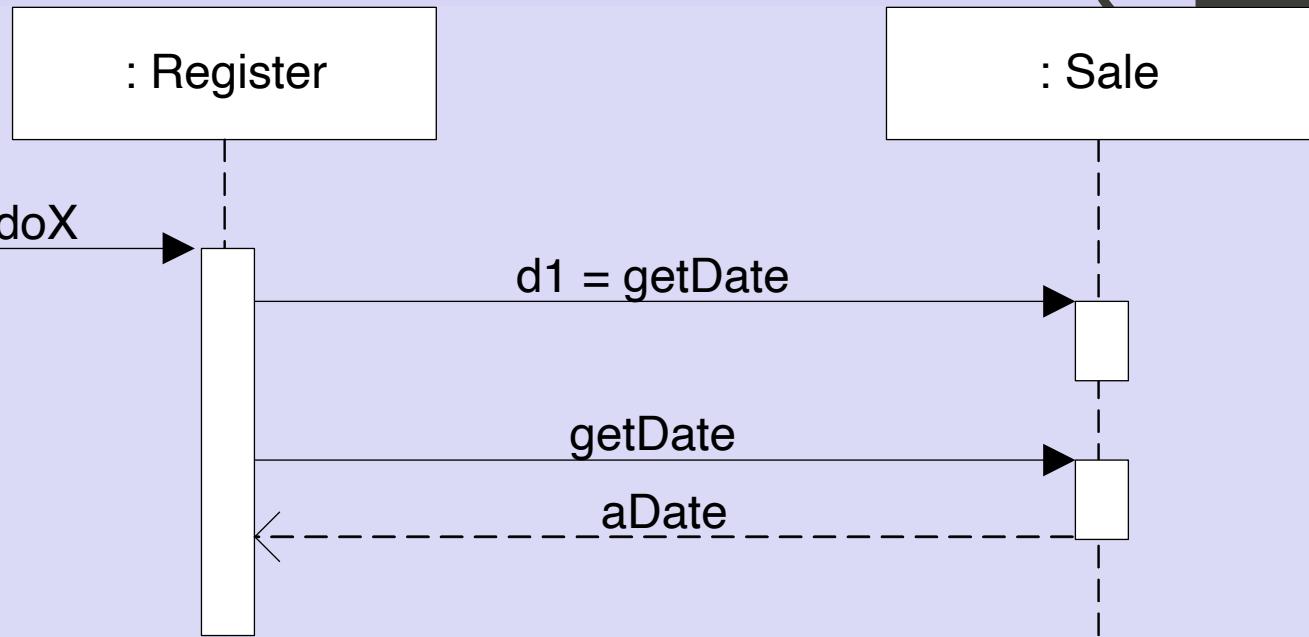
:Vente - - - Instance

v:Vente - - - Instance nommée

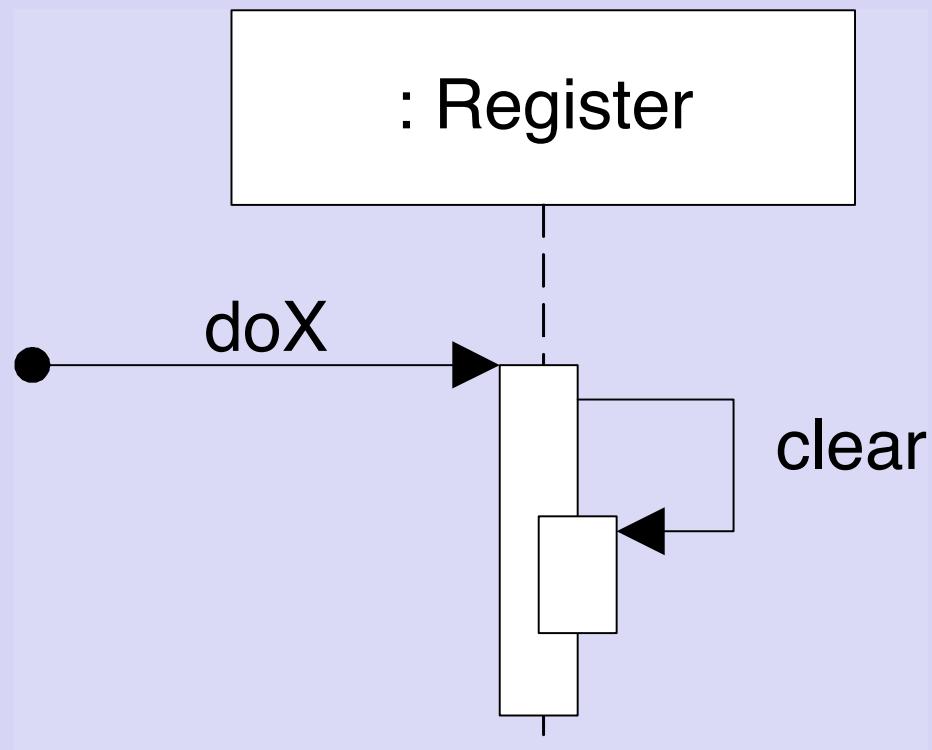
NOTATION: DIAGRAMMES DE SÉQUENCE



RÉSULTAT D'UN MESSAGE

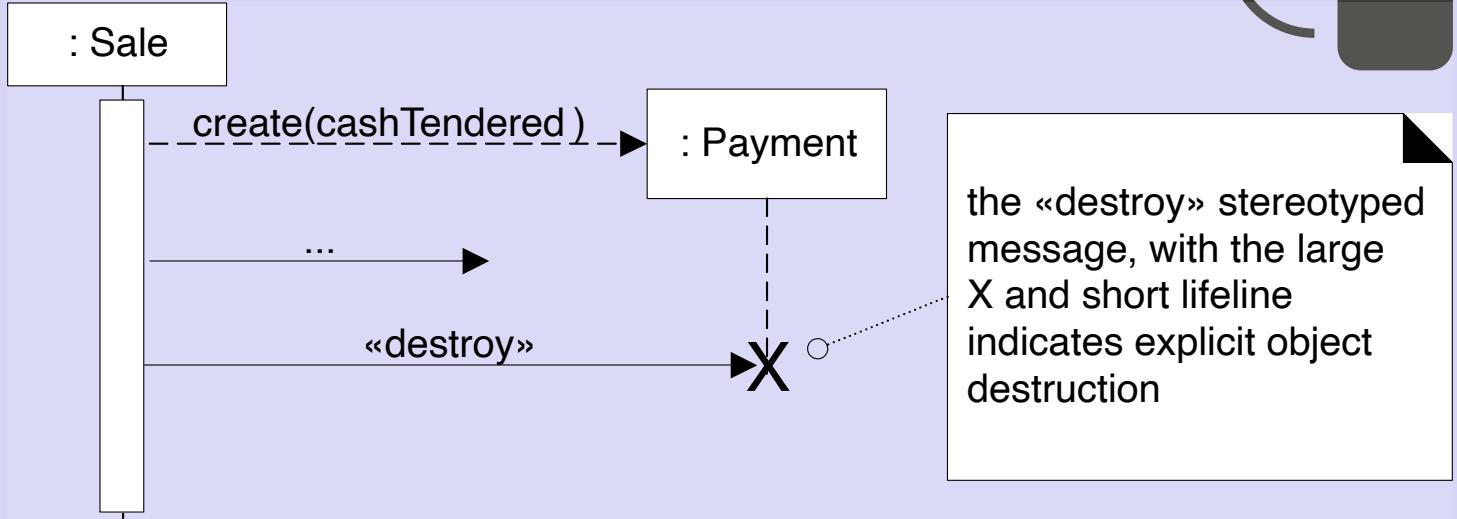
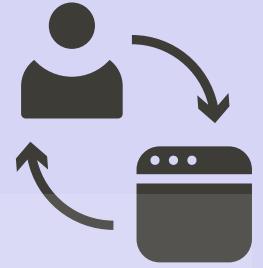


MESSAGE D'UN OBJET À LUI-MÊME



25 . 8

DESTRUCTION D'UN OBJET



ITÉRATION SUR UNE COLLECTION (NOTATION EXPLICITE)

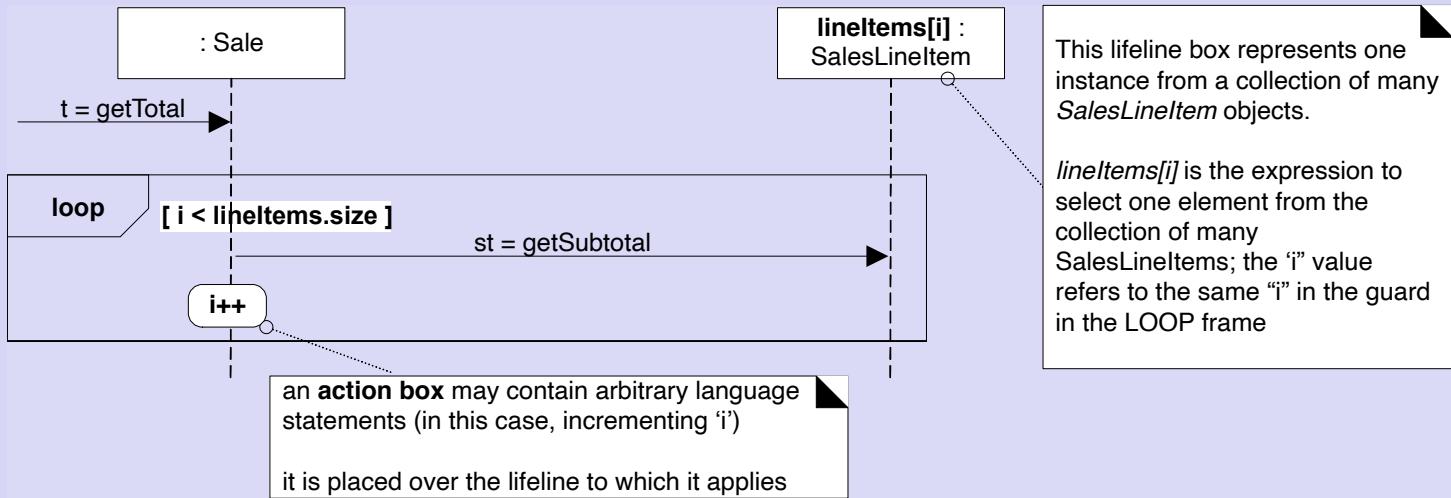
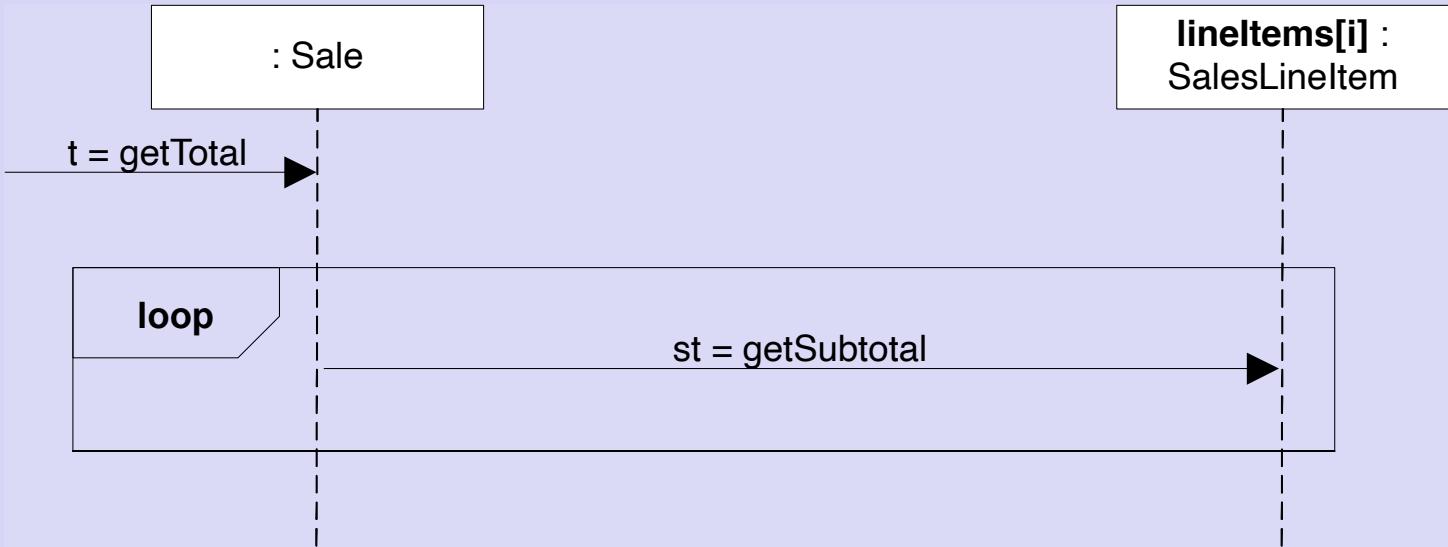


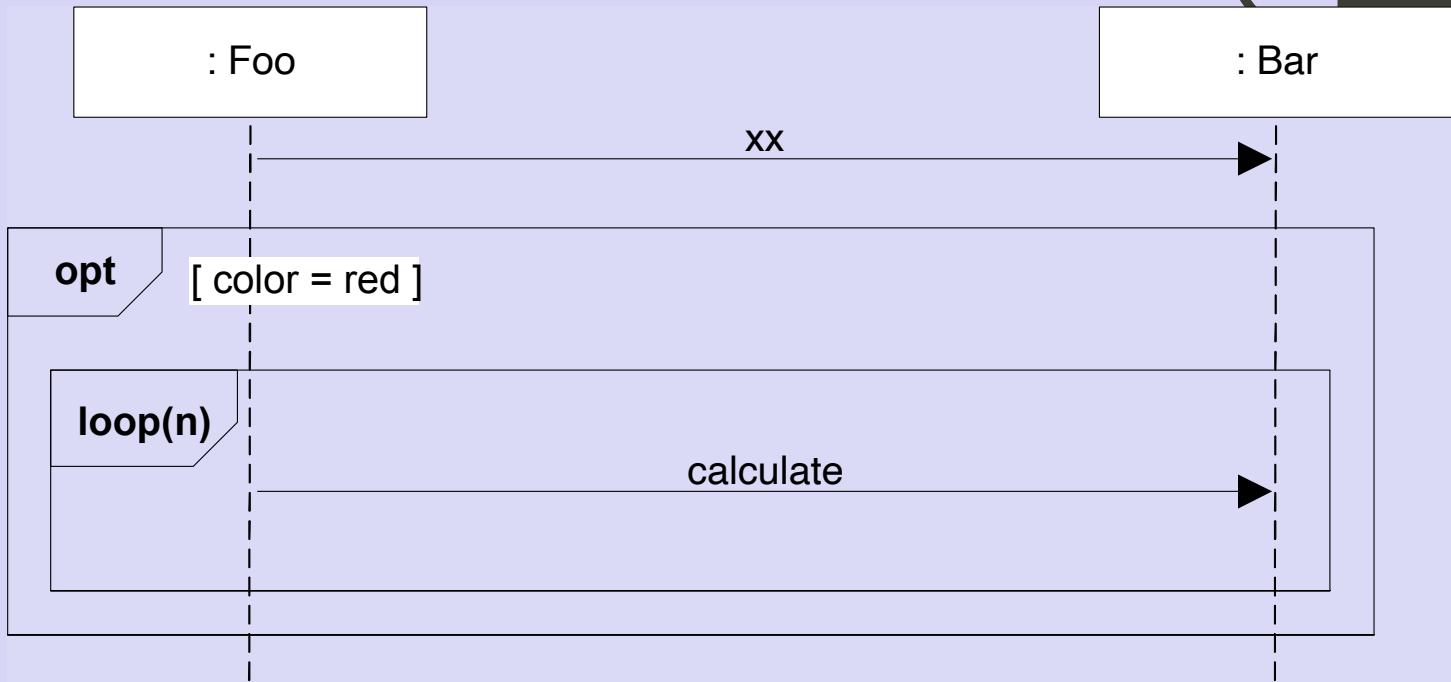
fig F14.16, A15.16

ITÉRATION SUR UNE COLLECTION (NOTATION IMPLICITE)



25 . 11

IMBRICATION DE CADRES



APPELS ASYNCHRONES ET OBJETS ACTIFS

a stick arrow in UML implies an asynchronous call

a filled arrow is the more common synchronous call

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

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

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

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

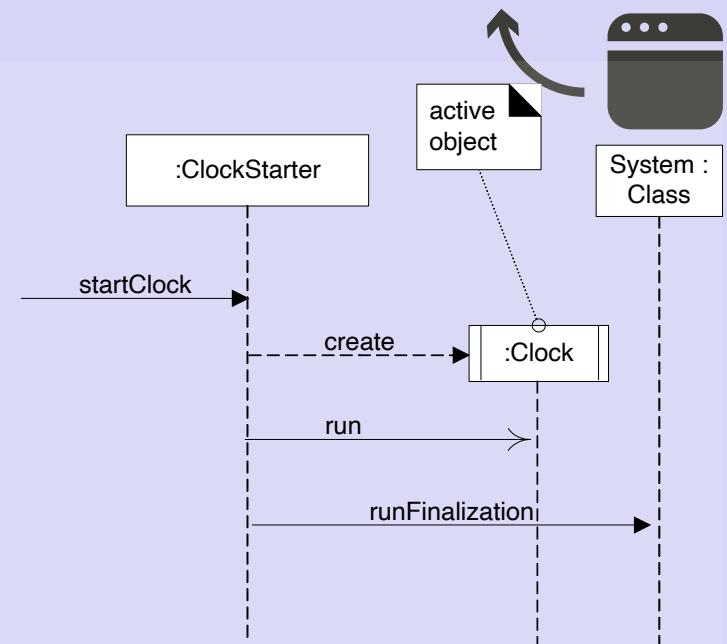
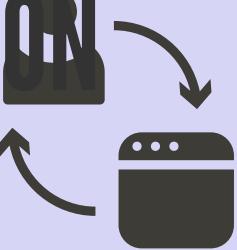


DIAGRAMME DE COMMUNICATION



- Liens et messages

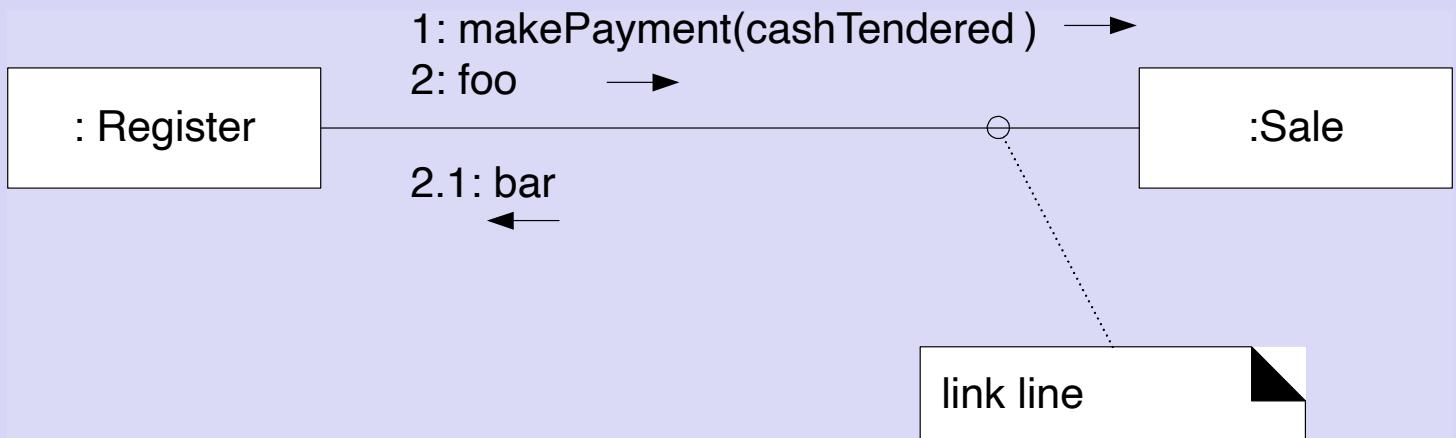
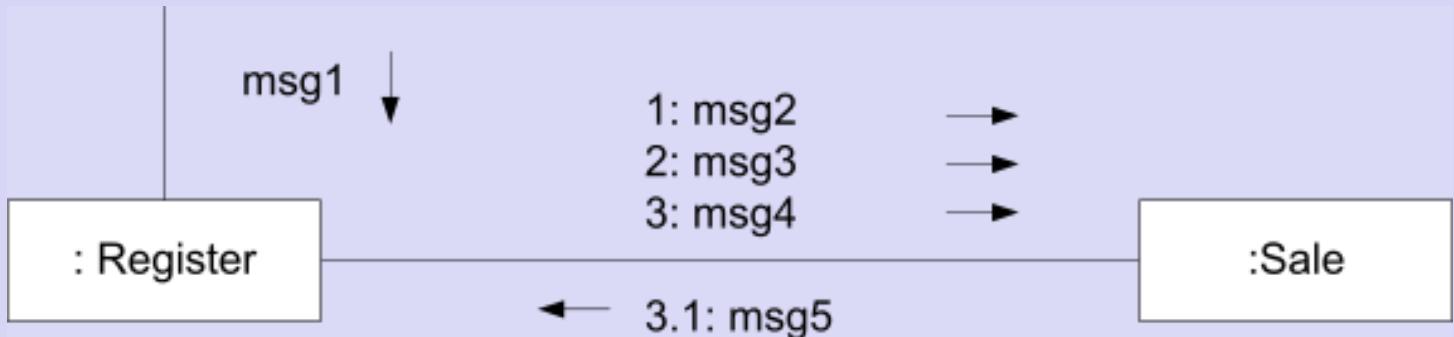


DIAGRAMME DE COMMUNICATION

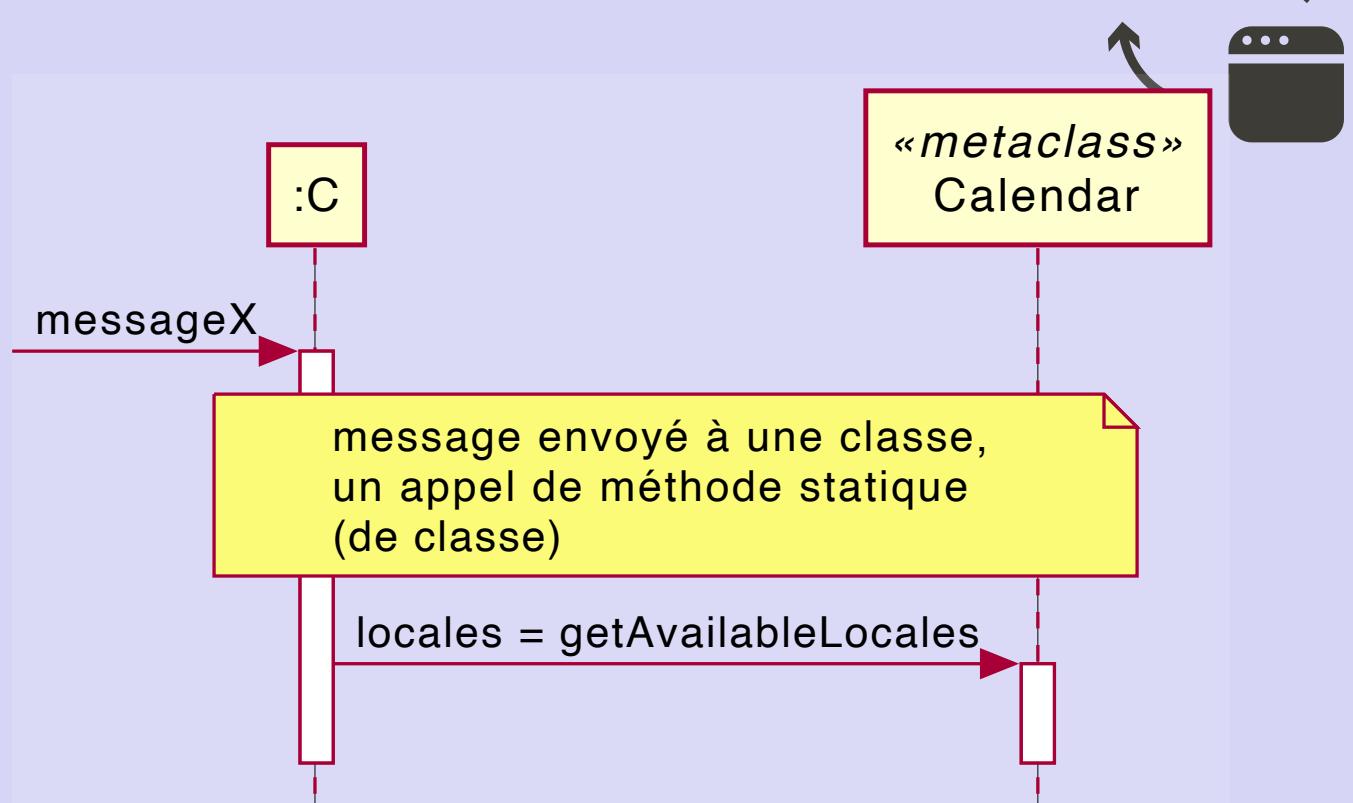


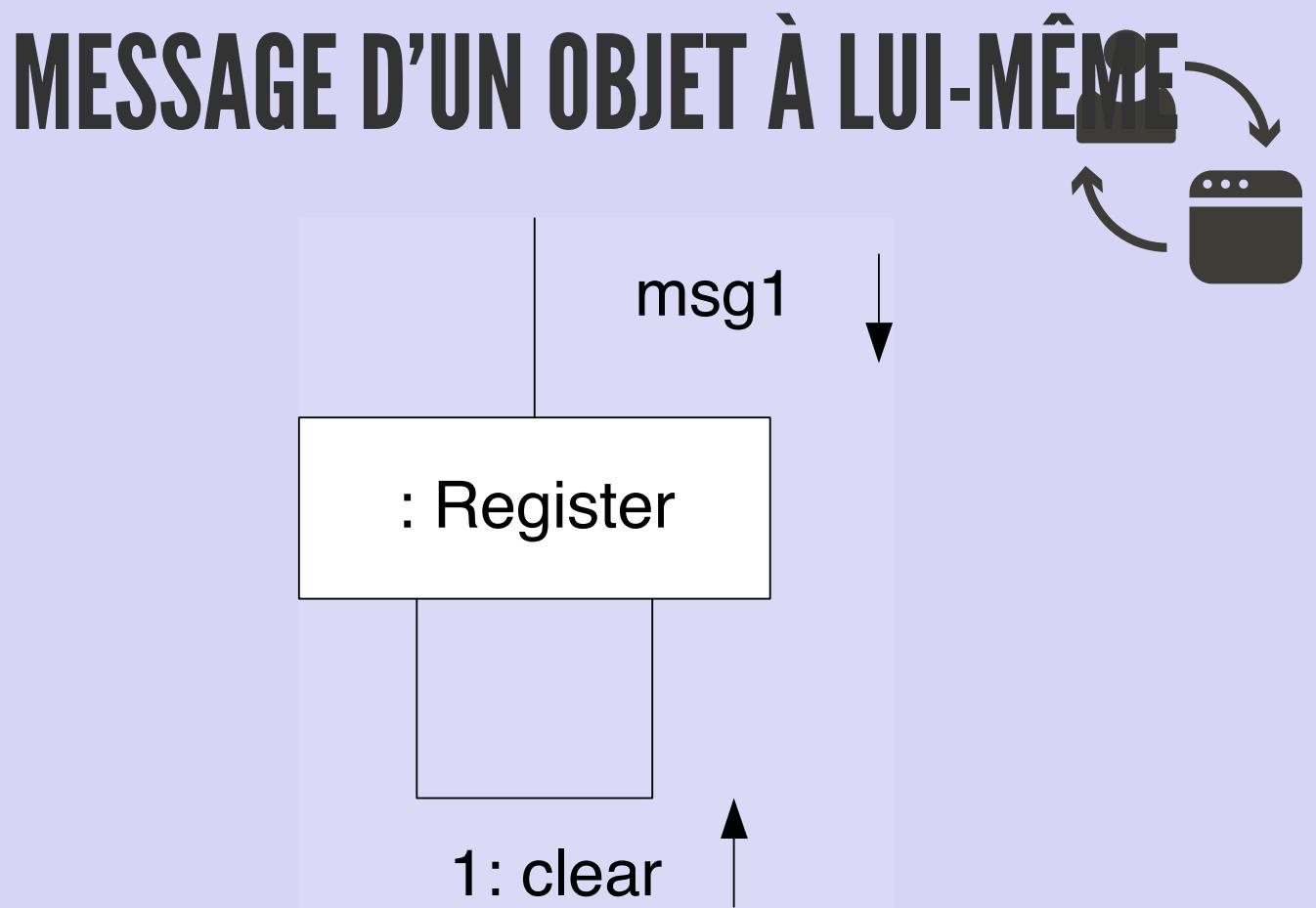
- Liens et messages



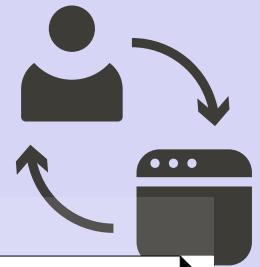
Note: Voir solution dans la section exercice plus bas.

MESSAGE POUR UNE CLASSE



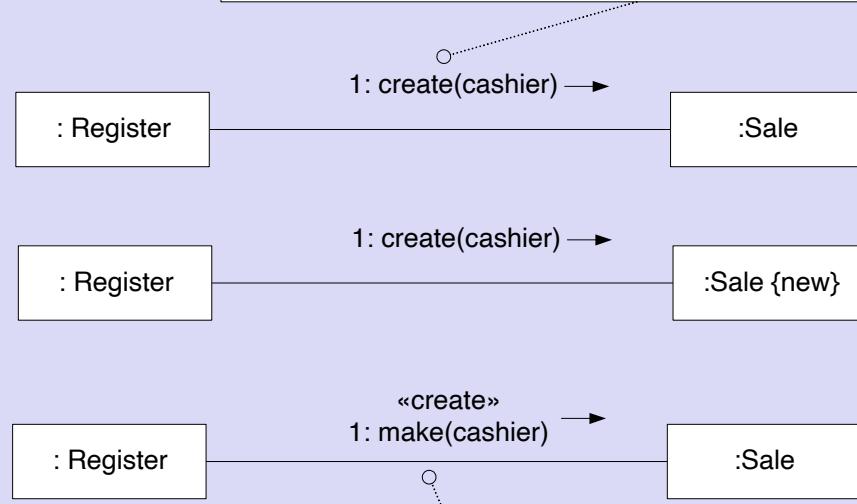


CRÉATION D'INSTANCE



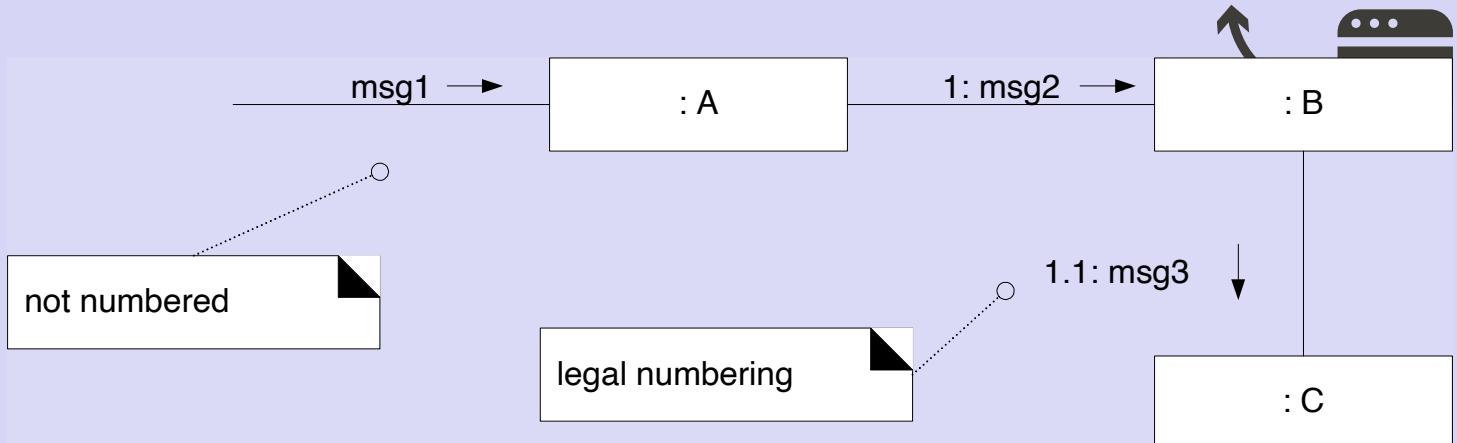
Three ways to show creation in a communication diagram

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



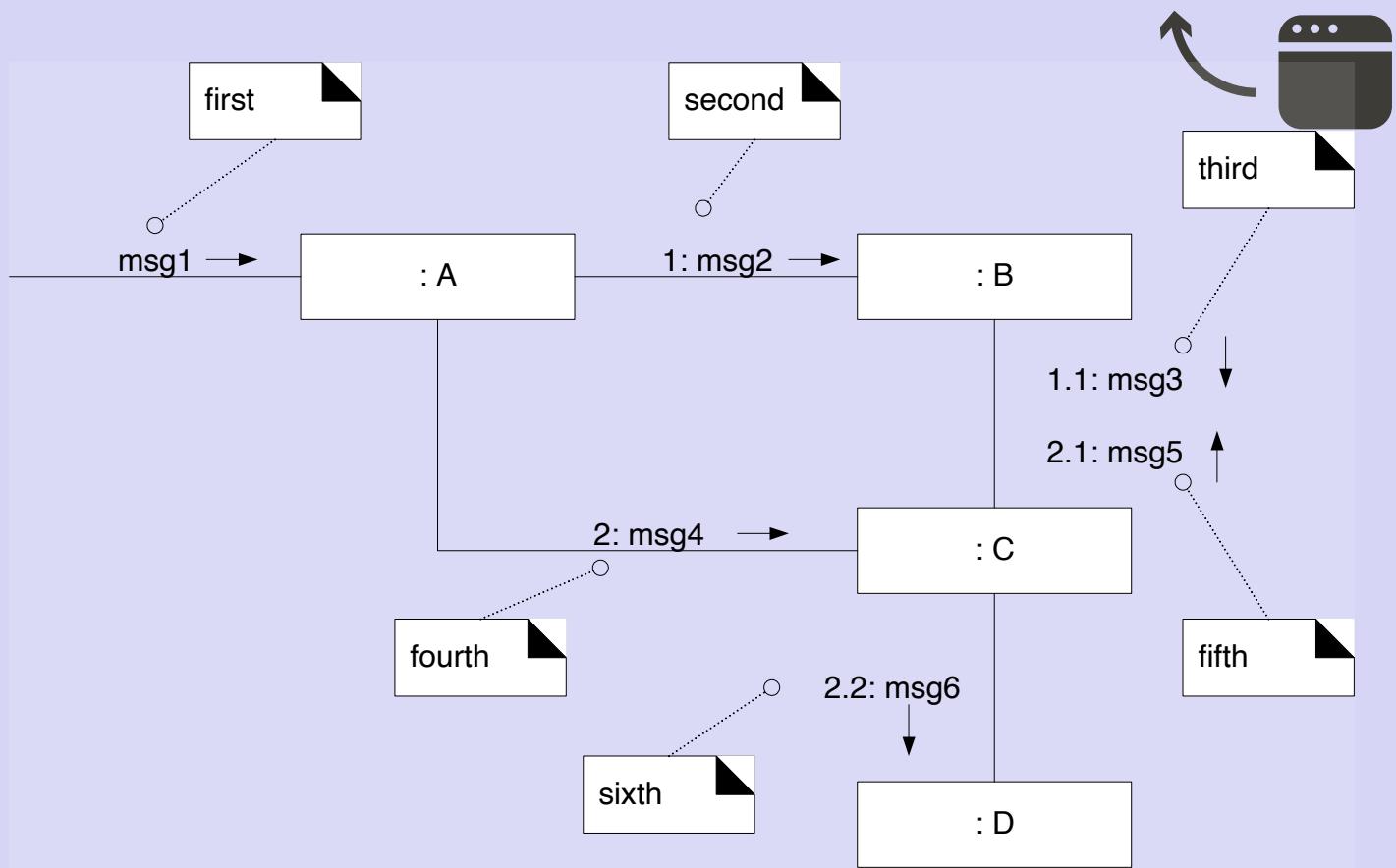
if an unobvious creation message name is used, the message may be stereotyped for clarity

AFFECTATION DE NUMÉROS D'ORDRE

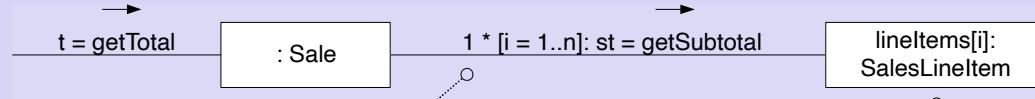


25 . 19

AFFECTATION DE NUMÉROS D'ORDRE



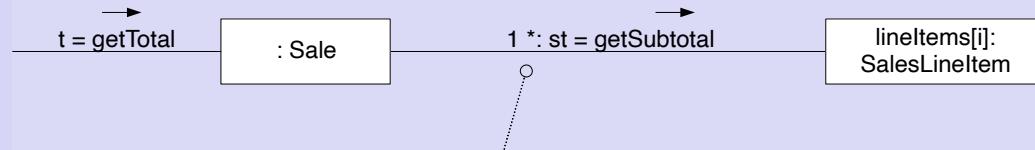
ITÉRATION SUR UNE COLLECTION



this iteration and recurrence clause indicates we are looping across each element of the `lineltems` collection.

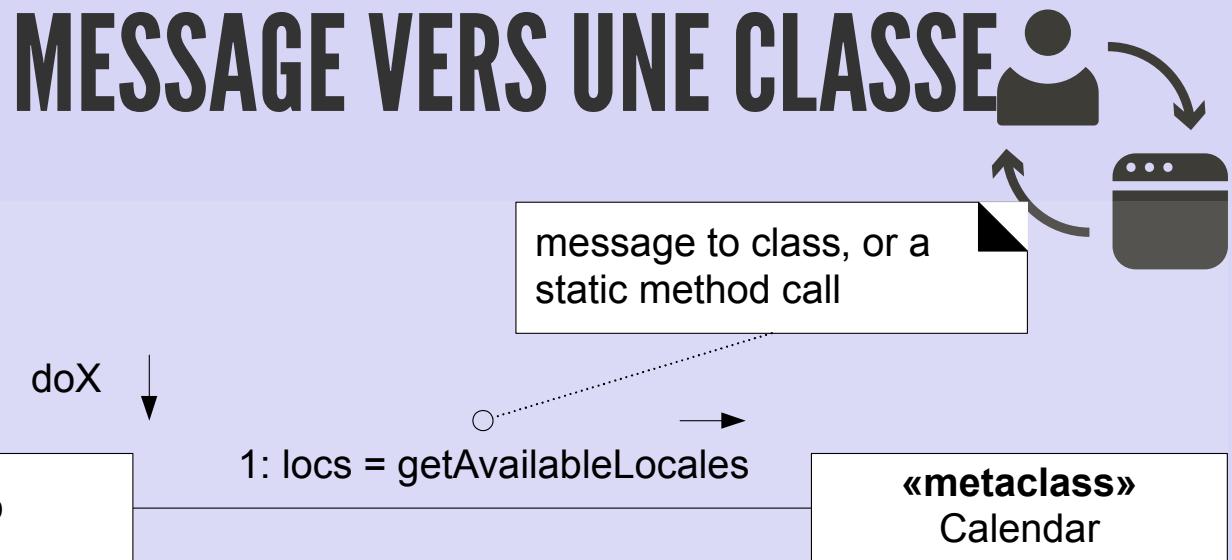
This lifeline box represents one instance from a collection of many `SalesLineItem` objects.

`lineltems[i]` is the expression to select one element from the collection of many `SalesLineItems`; the 'i' value comes from the message clause.

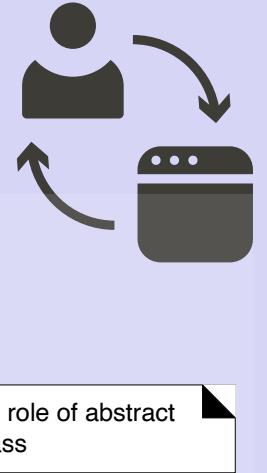


Less precise, but usually good enough to imply iteration across the collection members

collection en java



MESSAGES POLYMORPHES



*EXERCICE - TRANSFORMER EN DIAGRAMME DE SÉQUENCE, DE CLASSES

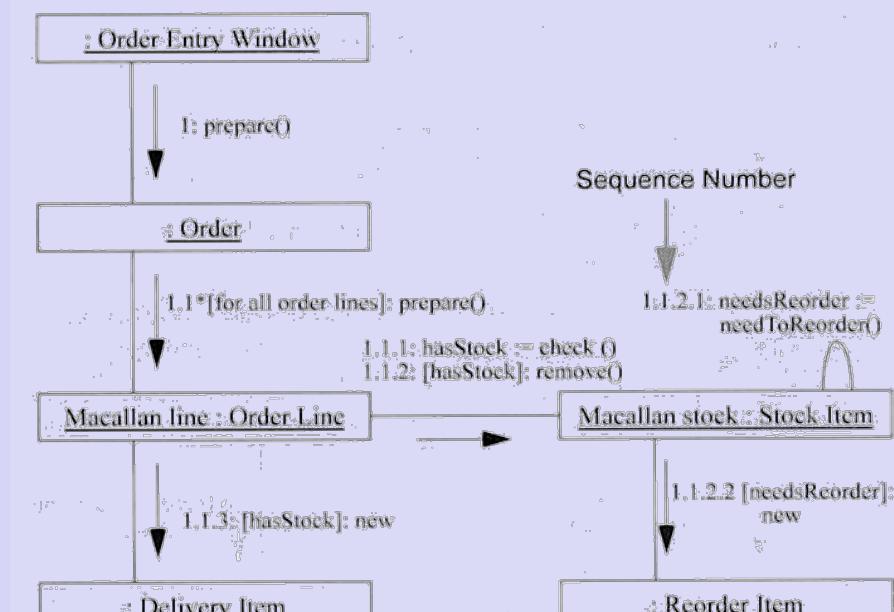


Figure 5-5: Collaboration Diagram with Decimal Numbering

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

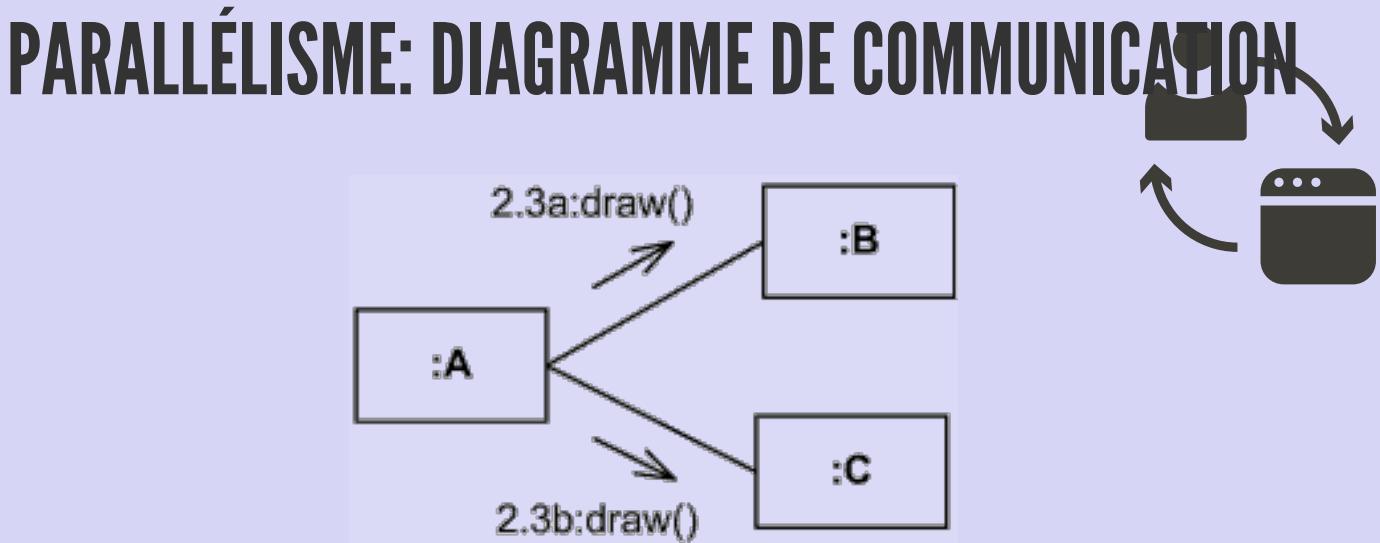
*EXERCICE

TRANSFORMER EN DIAGRAMME DE SÉQUENCE



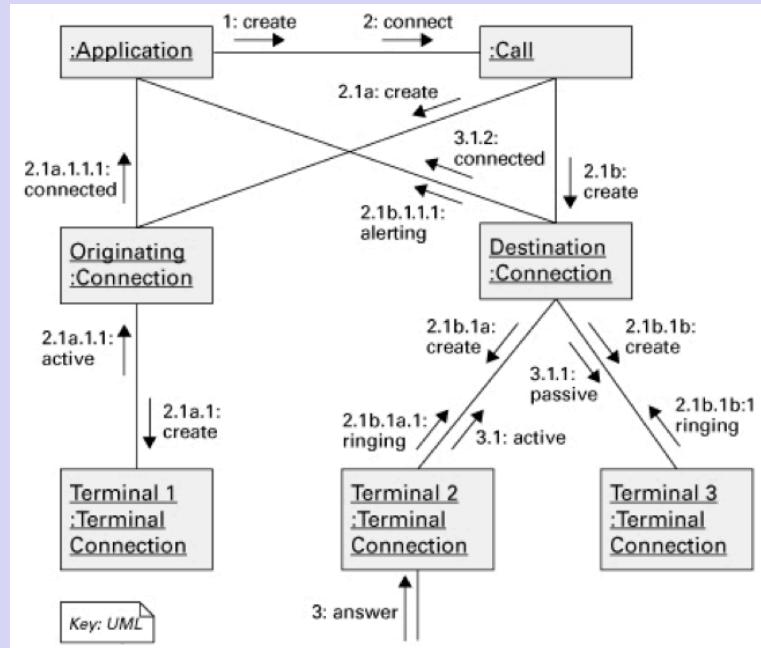
larman/F14.24, A15.24

25 . 25



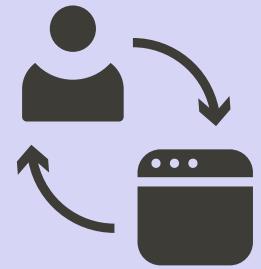
- For example,
 - messages 2.3a and 2.3b are concurrent within activation 2.3,
 - Name represents a concurrent thread of control.
 - Instance of A sends draw () messages concurrently to instance of B and to instance of C
- Réf: <https://www.uml-diagrams.org/communication-diagrams.html>

DIAGRAMME DE COLLABORATION UML

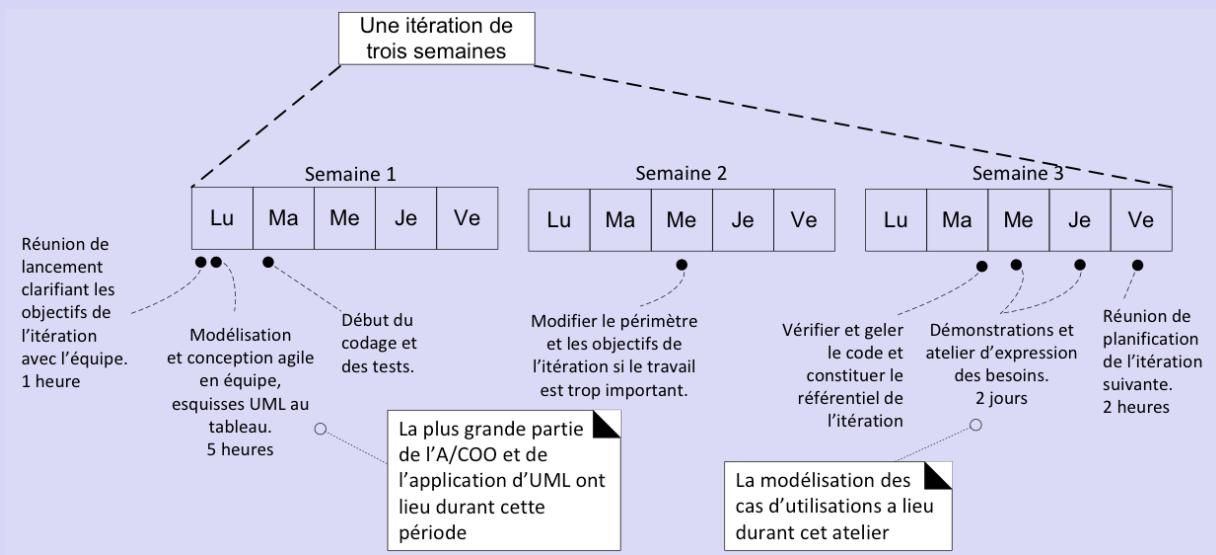


Réf: Documenting Software Architectures: Views and Beyond, 2e édition, Felix Bachmann, Len Bass, Paul C. Clements, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith A. Stafford

RÉSUMÉ



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



MÉTHODOLOGIE DE DÉVELOPPEMENT UML

- UML est un langage de modélisation général qui inclut un large éventail de concepts, notations et diagrammes.
- Avant d'intégrer UML dans un processus de développement , il est fondamental de définir correctement une méthodologie pour prendre en charge son utilisation.
 - La définition d'une méthodologie de développement facilitera l'intégration et l'adoption d'UML , augmentera la qualité des modèles et augmentera la productivité des développeurs.

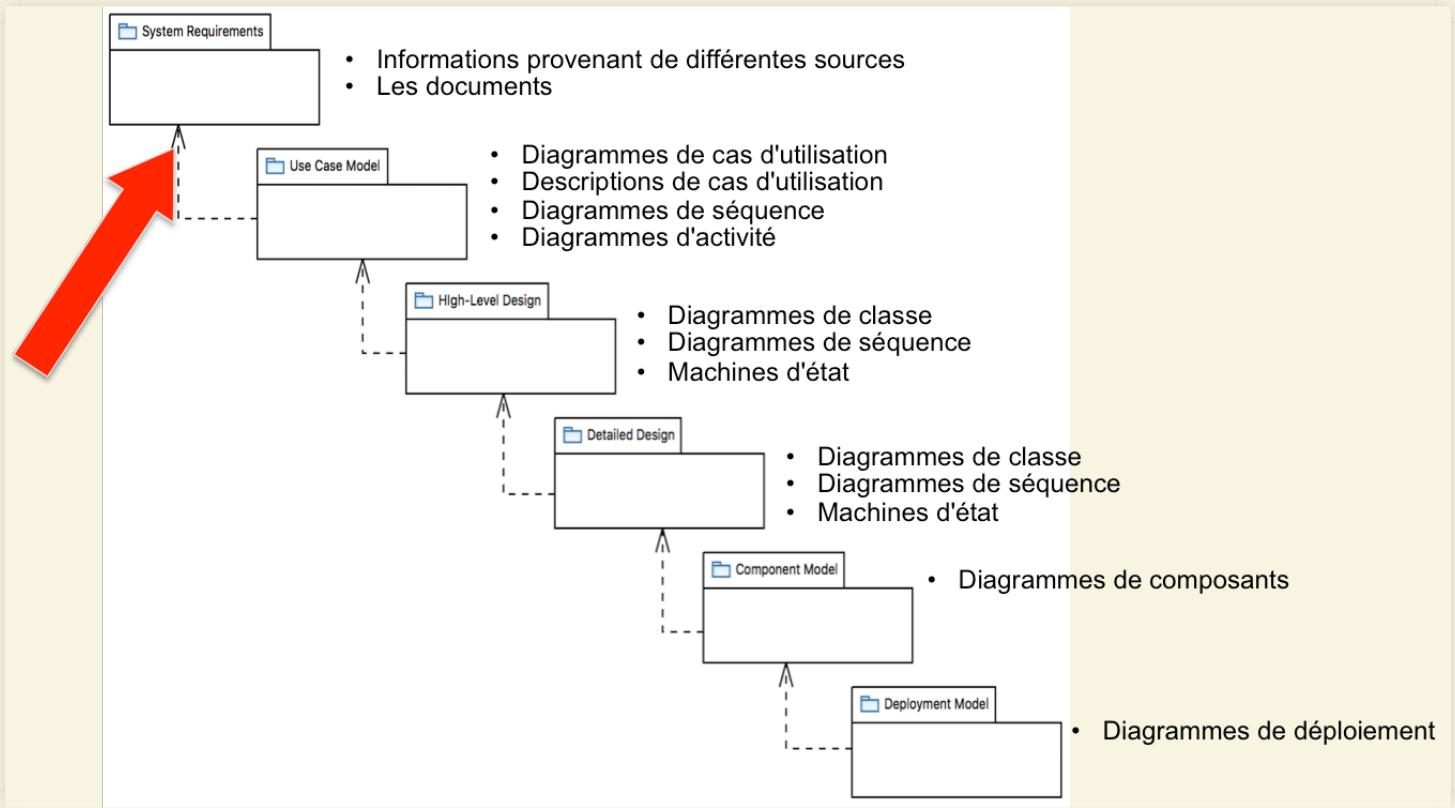
MÉTHODOLOGIE DE DÉVELOPPEMENT UML

- Pour chaque phase de développement , vous devez définir
 - Ensemble de diagrammes UML utilisés dans la phase
 - Rôle de chaque diagramme
 - Sous-ensemble de la notation UML utilisée
 - Remarque: certains diagrammes peuvent jouer différents rôles dans différentes phases du processus de développement .
 - Par exemple . Les diagrammes de séquence peuvent être utilisés pour décrire les interactions entre le système et les acteurs lors de la phase de modélisation de cas d'utilisation , fournir des détails sur l'exécution de scénarios lors de la phase de conception et capturer des traces d'exécution lors de la phase de test.
- Définir un ensemble de règles et de directives pour développer des éléments de modèle aux différentes phases de développement

PERSONNALISER / CUSTOMISER UML

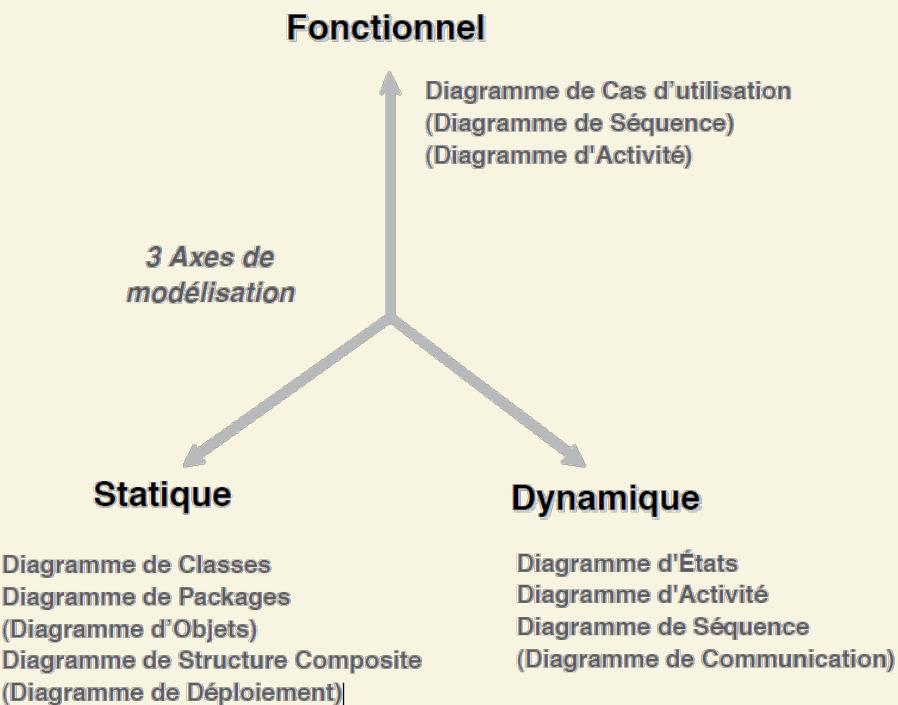
- UML fournit un ensemble de concepts, notations et diagrammes pouvant être utilisés dans un large éventail de contextes de développement.
- Cependant, seul un sous-ensemble de ces éléments est requis dans tout contexte de développement
- Le mécanisme de profil UML peut être utilisé pour personnaliser le langage UML pour des domaines de développement spécifiques.
- Mais la plupart des outils UML n'ont pas réussi à fournir un support de premier ordre pour la personnalisation et la modélisation spécifique à un domaine.

ÉTAPES DE MODÉLISATION

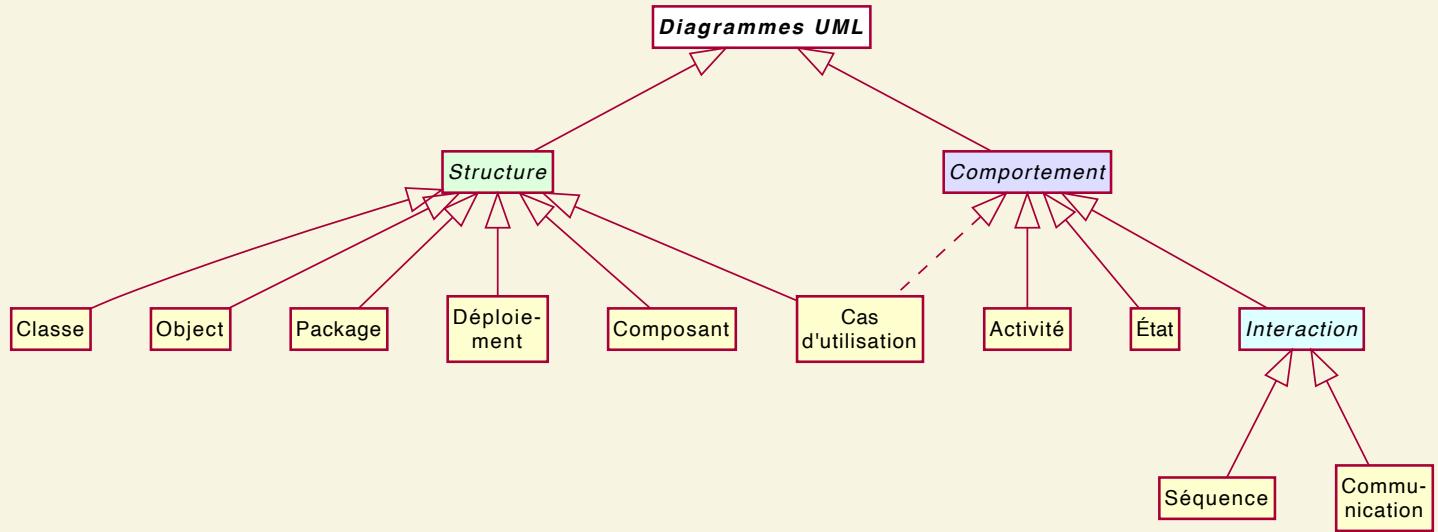


DIMENSIONS DE LA MODÉLISATION UML

Perspective selon UML par la pratique



STRUCTURE VS COMPORTEMENT



25 . 34

LES MEILLEURES PRATIQUES – CAS D'UTILISATIONS

- Développer Itérativement
- Utilisez la décomposition et le packaging pour structurer votre modèle de cas d'utilisation
- Définir l'ensemble des acteurs
- Définir / adopter un ensemble de guidelines et une convention de nommage
- Utiliser un modèle de description de cas d'utilisation «standard»
- Éloignez-vous de trop de détails , vous en parlerez plus tard
- Remarques:
 - Rappelez-vous que l'un des objectifs principaux des cas d'utilisation est de communiquer avec les parties prenantes , qui incluent des personnes de divers horizons.

Pas de solution magique , écrire des cas d'utilisation est un art!

LES MEILLEURES PRATIQUES – STRUCTURE

- Respecter le décalage de représentation
- Commencez par définir les classes directement dérivées des exigences et du modèle du domaine. Ajoutez de manière itérative de nouvelles classes et raffinez -les selon vos besoins au fur et à mesure de votre progression dans le processus de conception
- Définissez les principaux attributs de la classe . Ajoutez de manière itérative de nouveaux attributs au fur et à mesure que vous avancez dans le processus de conception de manière itérative
- Utiliser une convention de nommage cohérente pour les classes et les attributs

LES MEILLEURES PRATIQUES – DIAGRAMME D’INTERACTION

- Se concentrer sur le comportement global du système et la communication entre le système et l'ensemble des acteurs
 - Ne commencez pas à prendre des décisions de conception à ce stade
- Concentrez-vous sur les principaux scénarios , n'essayez pas d'être exhaustif
- Décomposer les diagrammes de séquences longs en fragments plus petits, le cas échéant
 - Définir des interactions séparées et utiliser « Interaction Use » pour faire référence à des interactions définies séparément

LES MEILLEURES PRATIQUES – DIAGRAMME ACTIVITÉ

- Choisissez le «bon» niveau d'abstraction / détails
 - Évitez le piège de spécifier les détails de conception,... vous le ferez plus tard!
- Maintenir la cohérence avec les descriptions de cas d'utilisation
 - Ne signifie pas une traçabilité individuelle
- Concentrez-vous sur la clarification et la compréhension du modèle

LES MEILLEURES PRATIQUES – “DIAGRAMME D’ÉTAT”

- Choisissez le «bon» niveau d’abstraction / détails
 - Évitez le piège de spécifier les détails de conception,... vous le ferez plus tard dans le processus de conception!
- Maintenir la cohérence avec le modèle de cas d’utilisation
 - Ne signifie pas une traçabilité formelle individuelle
- Exploiter les relations de cas d’utilisation (et de scénarios) pour structurer les machines d’états
- Ne définissez pas une machine d’états pour chaque classe , mais uniquement pour les classes actives

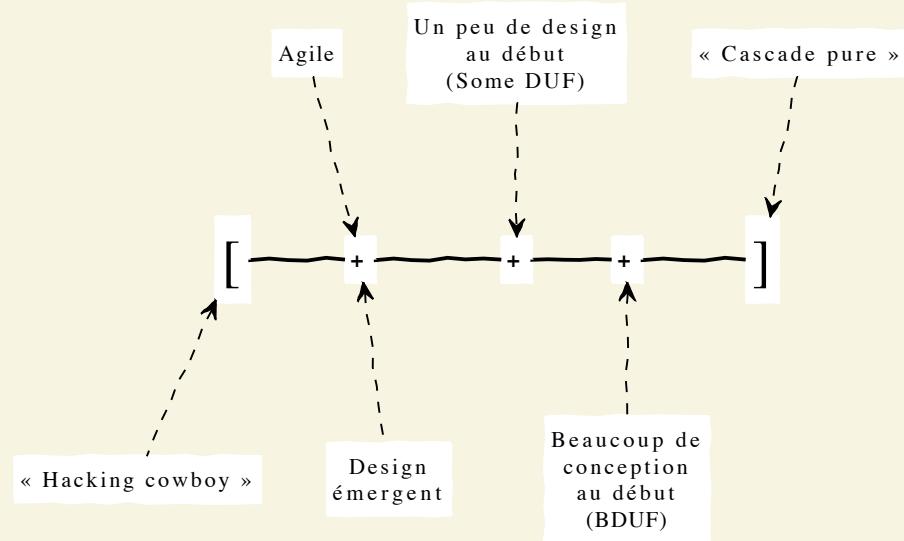
MODULE DETTE TECHNIQUE

2. Dette technique - 51.06m
3. Dette technique vscode- 4.09m

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

DETTE TECHNIQUE

Rappel : Spectre de la conception de Neal Ford.



DETTE TECHNIQUE : RISQUE

La dette est une forme de risque.

Elle apporte des *bénéfices* ou des *pertes*, selon la quantité d'intérêt à payer.

Chapitre 18 des notes de cours

DETTE TECHNIQUE: MÉTAPHORE

Le “Hacking Cowboy” 😎 (s’endetter)

- Bénéfice: obtenir une solution rapidement
- Perte: utiliser ce code à long terme

Pour *rembourser* la dette, on doit réécrire le code (faire un meilleur design, écrire des tests, “refactor”, etc.)

DETTE TECHNIQUE: EXEMPLE

Accumulation de décision dans le temps qui donne un code de mauvaise qualité et constraint la solution architecturale

- Conséquences
 - Taux élevé de défaut
 - Retard du projet
 - Complication d'entretien
 - La productivité de développement très faible

- Proxy pour dette techniques
 - Combien de temps faut t'il pour mettre à jour le système
 - Combien coutent la mise à jour du système.
 - Combien de fois les choses sont cassées.

CLASSIFICATION DE LA DETTE TECHNIQUE

Fowler, 2009

| Dette | Imprudente | Prudente |
|--------------|--|---|
| Délibérée | “On n'a pas le temps pour la conception” | “Faut livrer maintenant puis en assumer les conséquences” |
| Involontaire | “C'est quoi la séparation en couches?” | “Maintenant on sait comment on aurait dû le faire” |

DETTE TECHNIQUE: AUTRE EXEMPLES

- Ne pas migrer à la dernière version du framework de développement
- Ne pas faire d'analyse
- Ne pas faire de conception
- Ne pas faire de test
- Ne pas avoir de tracabilité entre les différents documents

5 ARGUMENTS POUR QUE LES MANAGERS SE SOUCIENT DE LA DETTE TECHNIQUE

1. Le réusinage réduira la volatilité du coût marginal des caractéristiques»
2. Au cours des 3 derniers mois, nous avons dépensé 63% du budget de développement pour régler des questions de qualité
3. Nous avons pris des prêts techniques pour expédier plus rapidement, nous devons rembourser une dette pour garder un délai de marché faible"
4. Nous pouvons réduire nos dépense d'affaires en investissant 10% de notre temps dans la qualité de notre code"
5. Investir 20% du budget dans le réusinage coupera le temps de première réponse de moitié avec un ROI positif sur la productivité de développeurs

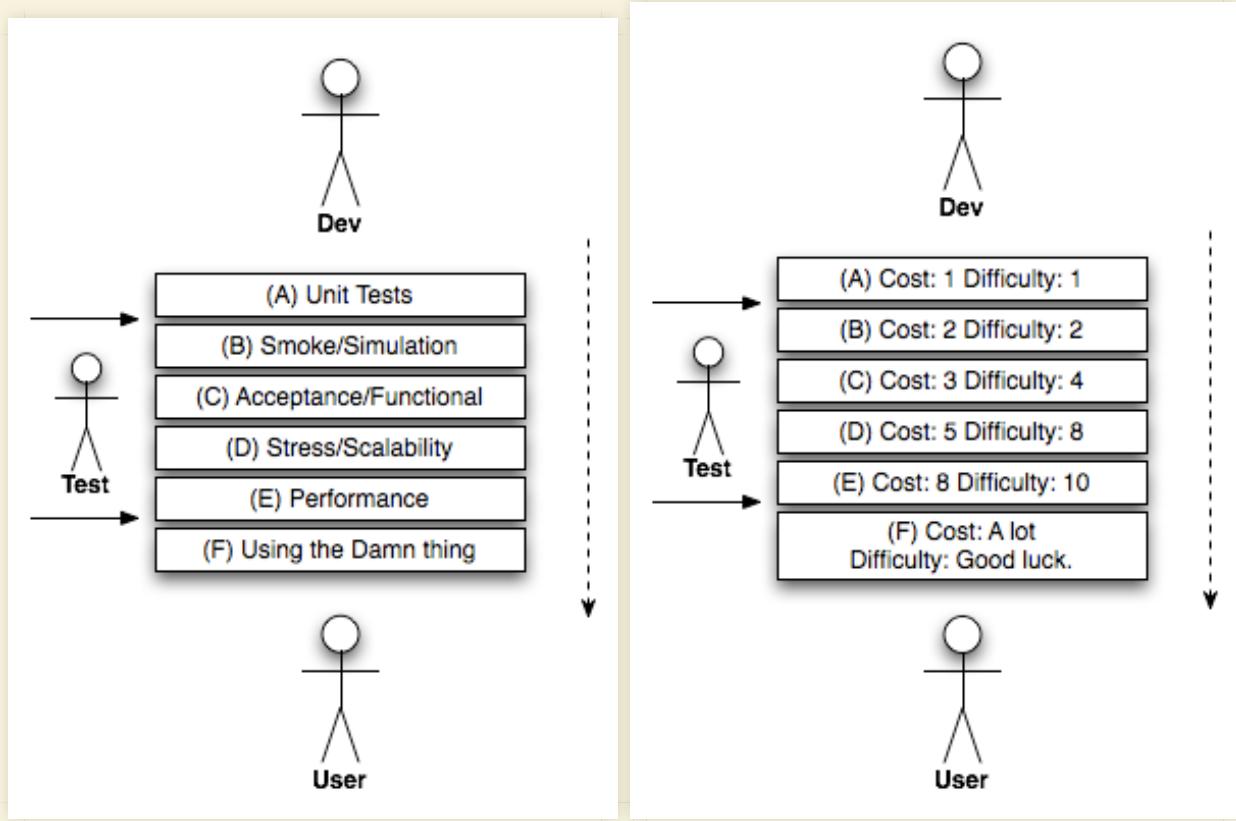
réf.: <https://understandlegacycode.com/blog/5-arguments-to-make-managers-care-about-technical-debt/>

DETTE TECHNIQUE: TESTS

- Essentiellement, il est dans votre intérêt, en tant que développeur, en tant qu'équipe, d'encourager de nombreux tests plus bas dans les piles présentées ici. Cela commence par des tests unitaires complets et vérifiés. Cela continue avec une discipline de test solide et reproductible (pour laquelle je recommande l'automatisation des tests).
- Pourquoi? Parce que - lorsque vous vous déplacez plus haut dans la pile, ce fichu bug que quelqu'un a enregistré est caché derrière couche après couche de code. Plus le bogue est éloigné du niveau d'unité, plus les composants et les variables d'environnement sont impliqués. Plus ceux-ci sont impliqués, plus il est difficile d'identifier et de réparer, et plus le coût est élevé.

<http://jessenoller.com/blog/2008/09/17/the-cost-of-not-testing-software>

DETTE TECHNIQUE: TESTS

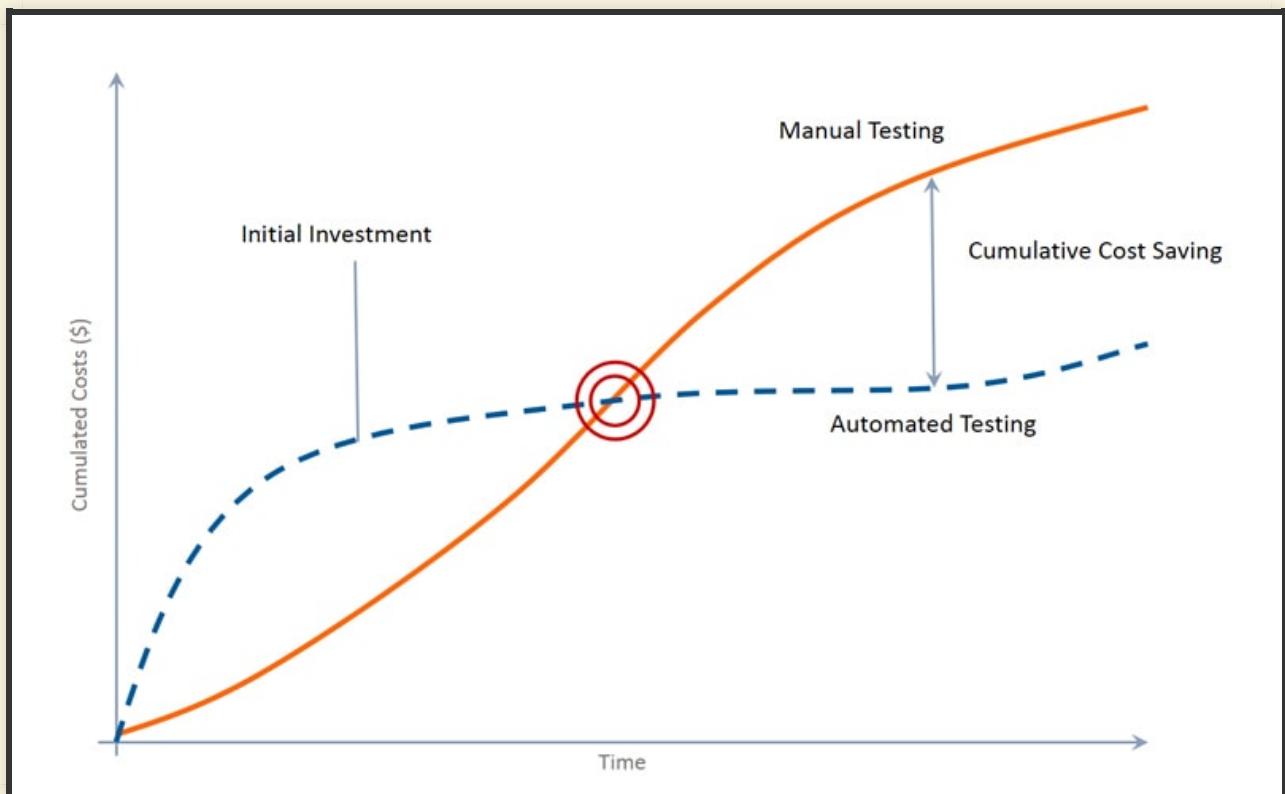


DETTES TECHNIQUE: TESTS

A 75\$/HEURE

| | Coding/Unit Testing | Integration | Beta Testing | Post-Release |
|------------------|---------------------|-------------|--------------|--------------|
| Hours to Fix | 3.2 | 9.7 | 12.2 | 14.8 |
| Cost to fix (\$) | 240 | 728 | 915 | 1,110 |

TEST MANUEL VS TEST AUTOMATISÉ



<https://abstracta.us/blog/test-automation/true-roi-test-automation/>

AVANTAGES

- Un testeur manuel exécute des tests 8 heures par jour et rentre chez lui. À ce moment, les tests s'arrêtent. Avec l'automatisation des tests, nous pouvons exécuter des tests 16 heures de plus par jour (dans le meilleur des cas... bien sûr) pour le même coût, ce qui réduit le coût moyen des heures de test.

| Manual | Automated |
|---|---|
| Hours $(10 \times 135) = 1,350 \text{ hours}$ | Hours $(3 \times 21 \times 16 \times 5) + (7 \times 135) = \text{Total of } 5985 \text{ hours}$ |
| Cost \$78/hour | Cost \$17.5/hour |

- Coût total : 101250\$/mois
- 3 développeurs x 21 jours/mois * 16h/jour de test * 5 fois plus rapide = 5040h équivalent
- 7 développeurs x 135h/mois = 945h

AVANTAGES

- Dans ce scénario, nous avons considérablement réduit le coût de chaque heure de test de 78 à 17,54, ce qui est un avantage que le directeur financier comprendra clairement. Ou vous pouvez le voir de cette façon; nous avons augmenté les tests de 1 350 heures à 5 985 heures équivalentes et gagné 315 000 \$ de tests par mois pour le même coût (5 040 fois le coût horaire moyen d'un testeur).

VALEUR COMMERCIALE

- Améliorez la qualité des logiciels
- Évitez les problèmes de fonctionnement
- Maintenir une bonne image client
- Évitez les problèmes juridiques
- Diminuer le coût de la correction des bogues de 5 fois

VALEUR INFORMATIQUE

- Simplifie les tâches de routine
- Exécutez plus de tests sans augmenter les coûts dans le même temps
- Augmentez l'étendue de la couverture
- Trouvez les défauts difficiles à détecter plus tôt, lorsqu'ils sont plus faciles à corriger
- Améliorez la qualité du logiciel

VALEUR INFORMATIQUE

- Simplifie les tâches de routine
- Exécutez plus de tests sans augmenter les coûts dans le même temps
- Augmentez l'étendue de la couverture
- Trouvez les défauts difficiles à détecter plus tôt, lorsqu'ils sont plus faciles à corriger
- Améliorez la qualité du logiciel

Êtes-vous d'accord pour dire que le retour sur investissement de l'automatisation des tests est important?

VISUAL STUDIO CODE

- voir plugin Tech Debt Metrics

MODULE DIVERS

1. Faute, erreur, échec et exception - 14.33m
2. Sondage sur technologies
3. Enquête sur la rémunération d'ingénieur... - 31.02m
4. Résautage
5. Stackoverflow developer Survey
6. Liste de recruteurs
7. Sales statistics
8. 97 things

[Séances](#) | [Quiz](#) | [Intro](#) | [Outils](#) | [ACOO](#) | [Équipe](#) | [PU](#) | [DCU](#) | [CU](#) | [IU](#) | [MDD](#) | [Dss](#) | [Contrat](#) | [RDCU](#) | [DCL](#) | [Couche](#) | [Exercice](#) | [Typescript](#) | [Complexite](#) | [FURPS](#) | [TDD](#) | [GOF](#) | [Diagramme d'état](#) | [Diagramme d'activité](#) | [Diagramme composant et déploiement](#) | [Diagrammes](#) | [Dette technique](#) | [Divers](#)

TOLÉRANCE AUX FAUTES

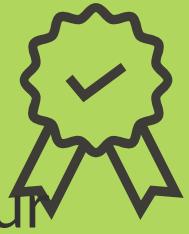


Created by Bharat
from the Noun Project

Définitions

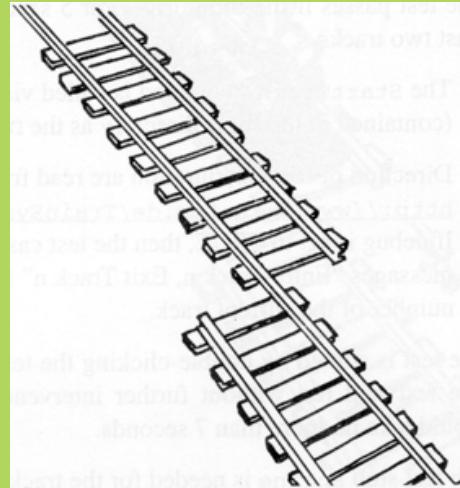
- Faute
 - source d'un mauvais comportement
- Erreur
 - manifestation de la faute dans le système en fonctionnement
- Échec (Défaillance)
 - service échoue à cause de l'erreur

FAUTE VS. ERREUR (1)



Faute ne produit pas toujours une erreur

Created by Bharat
from the Noun Project



ref: « Object-Oriented Software Engineering » BerndBruegge, B.Bruegge, AllenDutoit

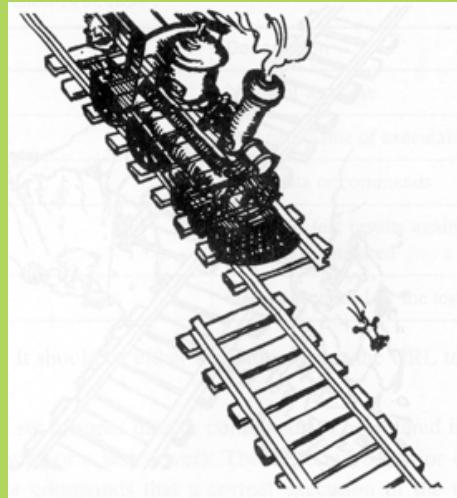
27 . 3

FAUTE VS. ERREUR (2)



L'erreur est la manifestation de la faute dans le système

Created by Bharat
from the Noun Project



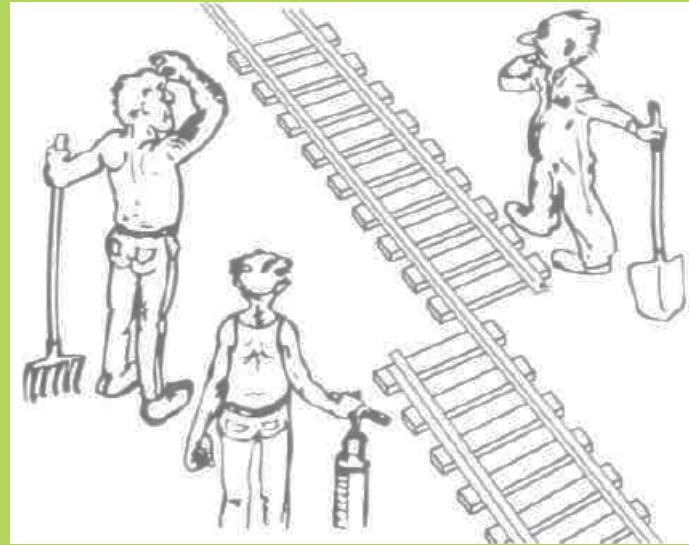
ref: « Object-Oriented Software Engineering » BerndBruegge, B.Bruegge, AllenDutoit

SOURCE D'UNE FAUTE

Problème algorithmique



Created by Bharat
from the Noun Project



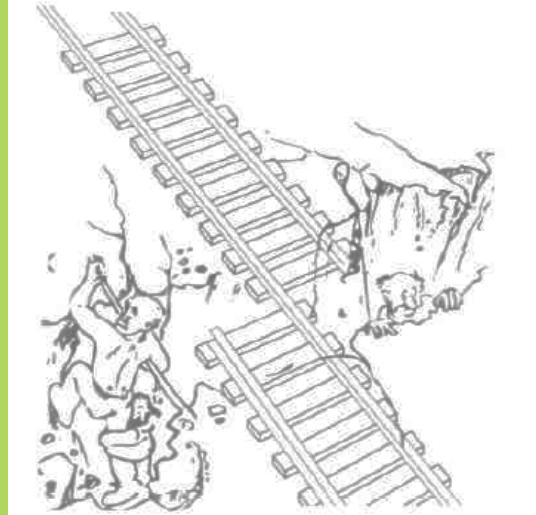
ref: « Object-Oriented Software Engineering » BerndBruegge, B.Bruegge, AllenDutoit

SOURCE D'UNE FAUTE

Problème environnemental



Created by Bharat
from the Noun Project



ref: « Object-Oriented Software Engineering » Bernd Bruegge, B. Bruegge, Allen Dutoit

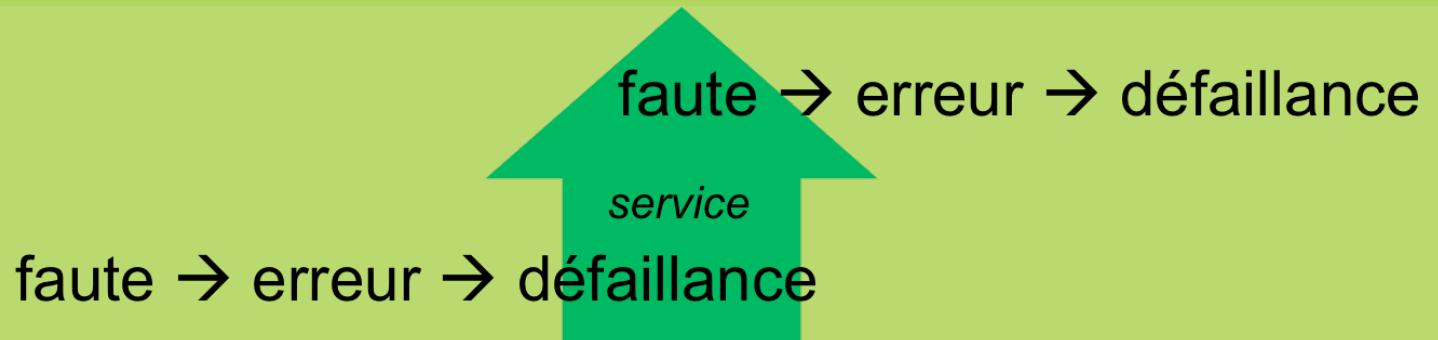
27 . 6

ENCHAÎNEMENT



Created by Bharat
from the Noun Project

- Une faute peut provoquer une erreur, qui peut provoquer une défaillance
- Vue d'un niveau plus haut, la défaillance peut être aperçue comme une faute



TRAITEMENT DES DÉFAILLANCES

- Quand rien ne va plus: *article n'est pas dans la cache et service web n'est pas accessible*
- Intervenants résolvent le problème: *entrée à la main du prix et de la description*

27 . 8

TRAITEMENT DES ERREURS </>

- Exceptions
- Utiles pour des ressources telles que
 - disque dur
 - mémoire
 - réseau
 - base de donnée
 - services externes

27 . 9

EXCEPTIONS: GRANDES LIGNES & />

- Patrons

- « Donner le nom du problème et non celui de l'objet qui lance l'exception »
- « Convertir exceptions »
- « Journal centralisé d'erreurs »
- « Dialogue d'erreur »

CONVERTIR EXCEPTIONS



- Comment présenter l'exception?
- Niveau d'abstraction
 - équivalent : faute -> erreur -> défaillance
(faute)
- Exception de haut niveau
 - encapsule exception de bas niveau

JOURNAL CENTRALISÉ D'ERREURS

- Objet singleton global
 - lui acheminer toutes les exceptions
 - flexibilité des définitions de flux de sortie
- `java.util.logging` (depuis JDK 1.4)

DIALOGUE D'ERREUR



- Objet singleton
 - mécanisme centralisé de notification des erreurs
 - pour notifier l'utilisateur
 - protection contre les variations dans le mécanisme de sortie
 - pas un objet de l'IU
 - indépendant de l'application

PATRON: DIALOGUE D'ERREUR

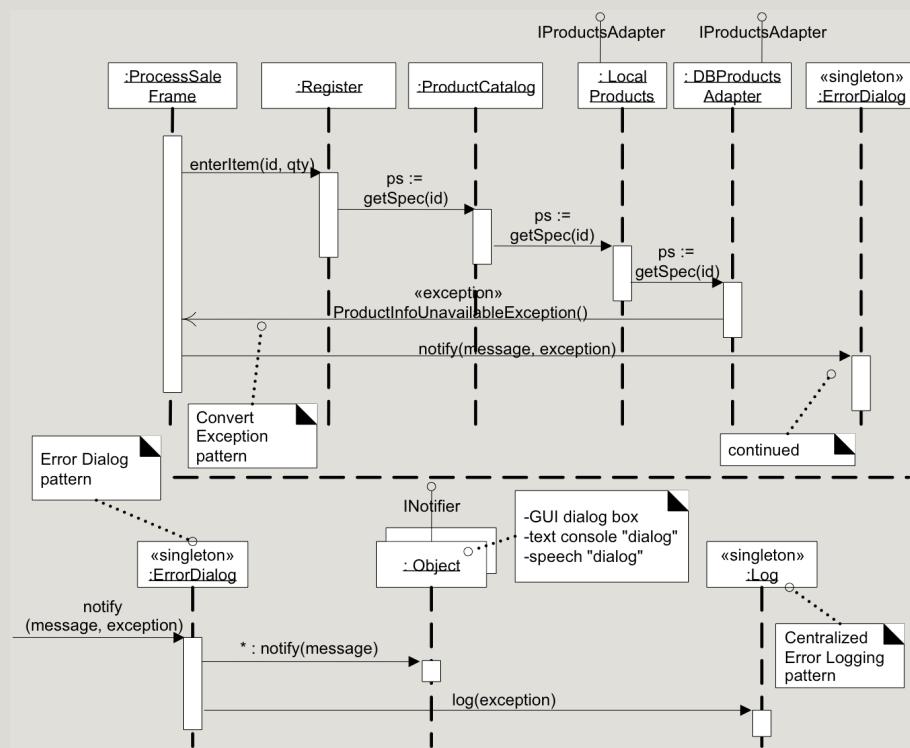


fig A35.11 p 592

PROBLÈME - SERVICE NON DISPONIBLE

- Dans le cas où on doit envoyer les ventes aux services de comptabilité, il faut agir le plus rapidement possible, donc directement communiquer avec les services de comptabilité externes.
- Dans l'éventualité où le service comptable est temporairement non disponible, comment peuvent-on éviter que cette faute devienne une défaillance.

Created by Bharat
from the Noun Project



Created by Jonathan Li
from the Noun Project

- Problème
 - l'accès direct à un objet est impossible (ou est indésirable). Que faire?
- Solution
 - Utiliser un objet proxy qui est un substitut à un objet
 - Le proxy implémente la même interface
 - Cela ajoute un niveau d'indirection

PATRON PROXY (GOF)



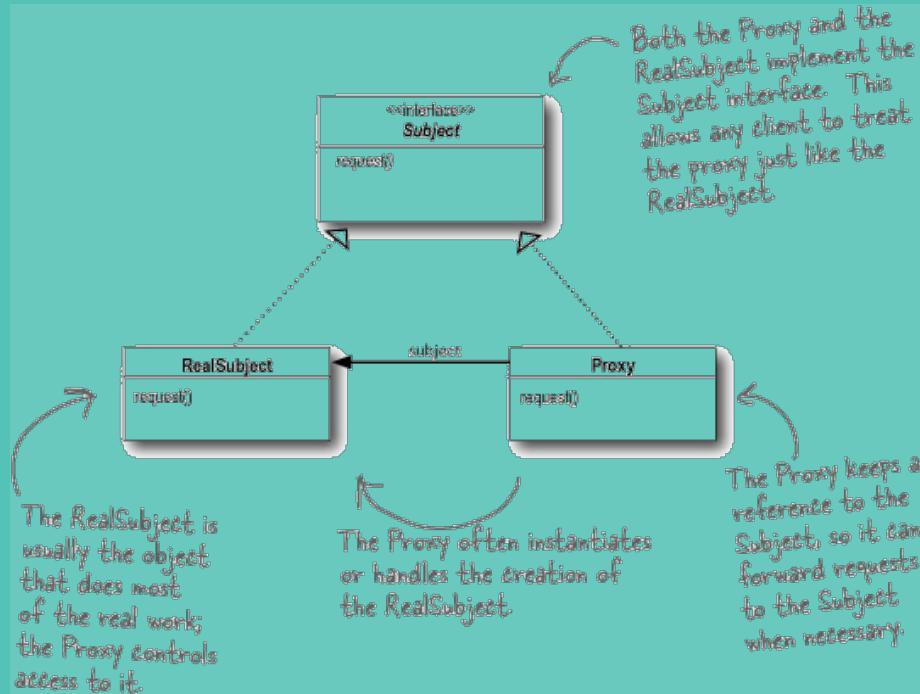
Created by Jonathan Li
from the Noun Project

- Problème
 - l'accès direct à un objet est impossible (ou est indésirable). Que faire?
- Solution
 - Utiliser un objet proxy qui est un substitut à un objet
 - Le proxy implémente la même interface
 - Cela ajoute un niveau d'indirection



Created by Jonathan Li
from the Noun Project

PROXY

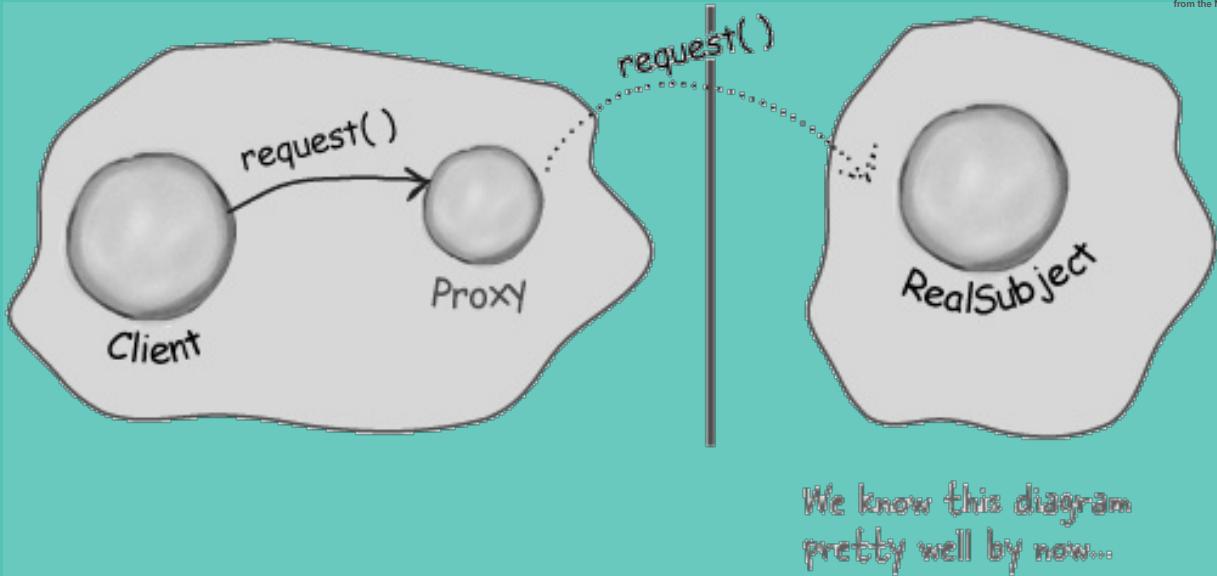


Freeman, Eric; Robson, Elisabeth; Bates, Bert; Sierra, Kathy. Head First Design Patterns.

PROXY



Created by Jonathan Li
from the Noun Project



Freeman, Eric; Robson, Elisabeth; Bates, Bert; Sierra, Kathy. Head First Design Patterns.



Created by Jonathan Li
from the Noun Project

PATRON PROXY (GOF)

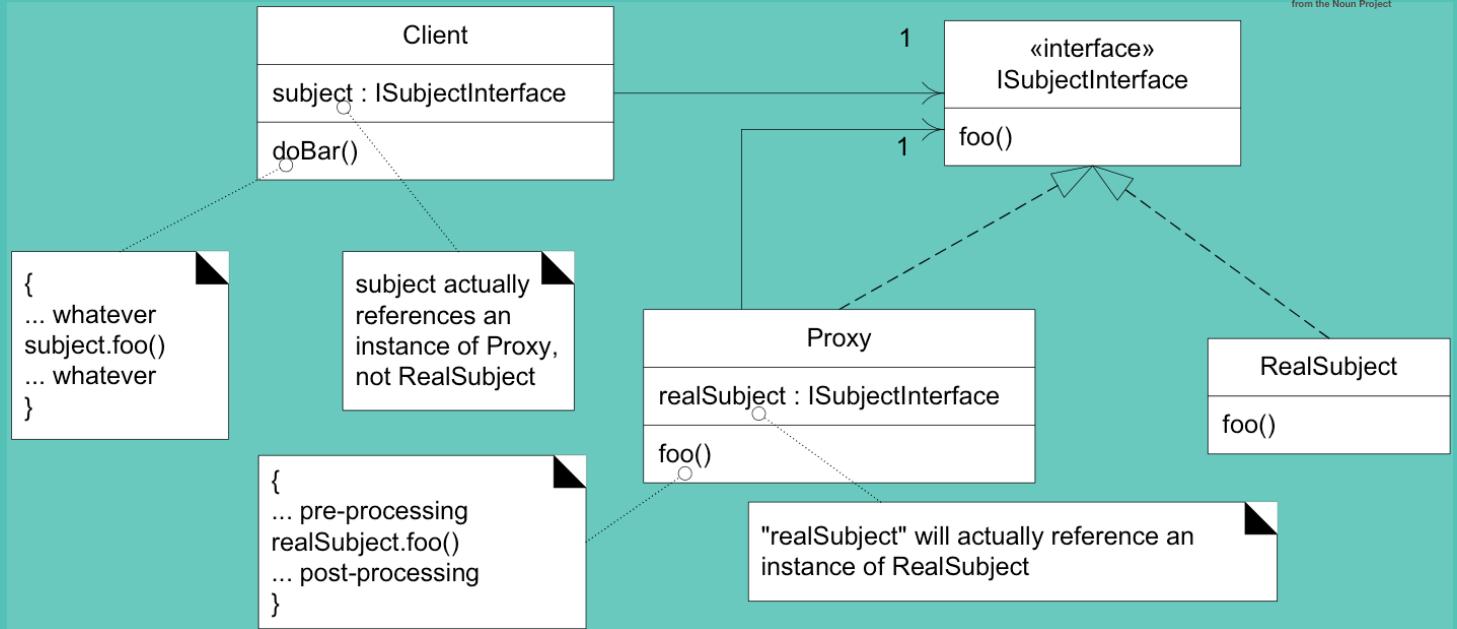
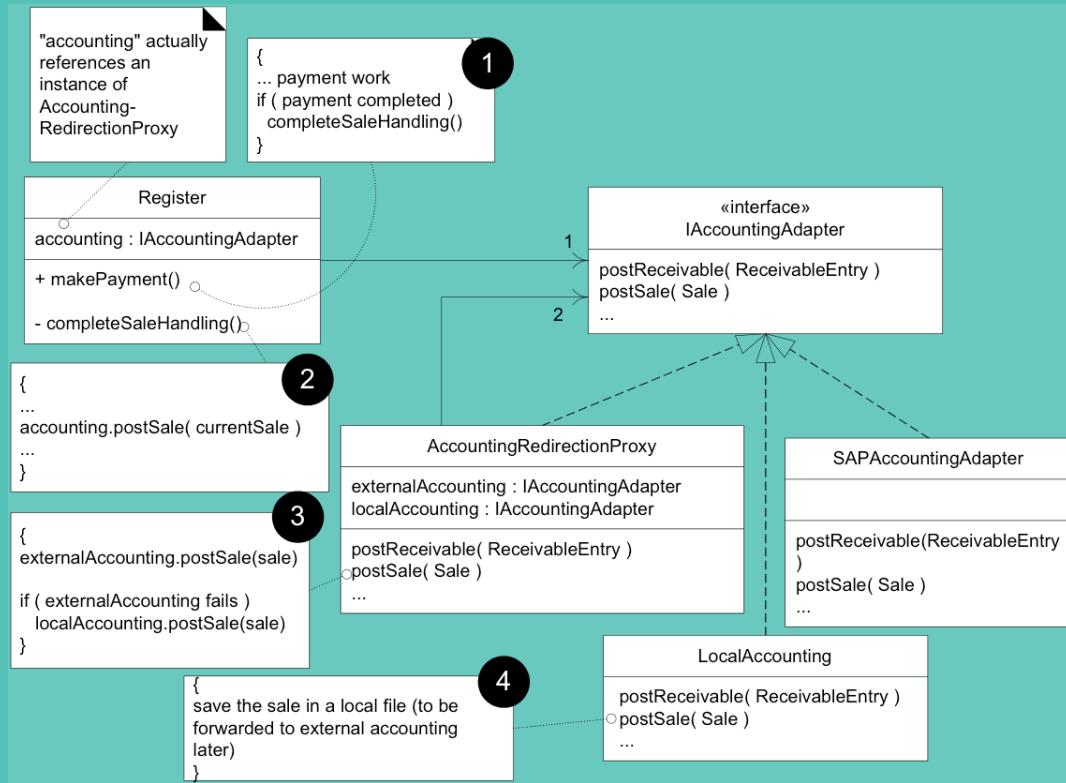


fig. F30.12

LES PROXY DANS POS



Created by Jonathan Li
from the Noun Project

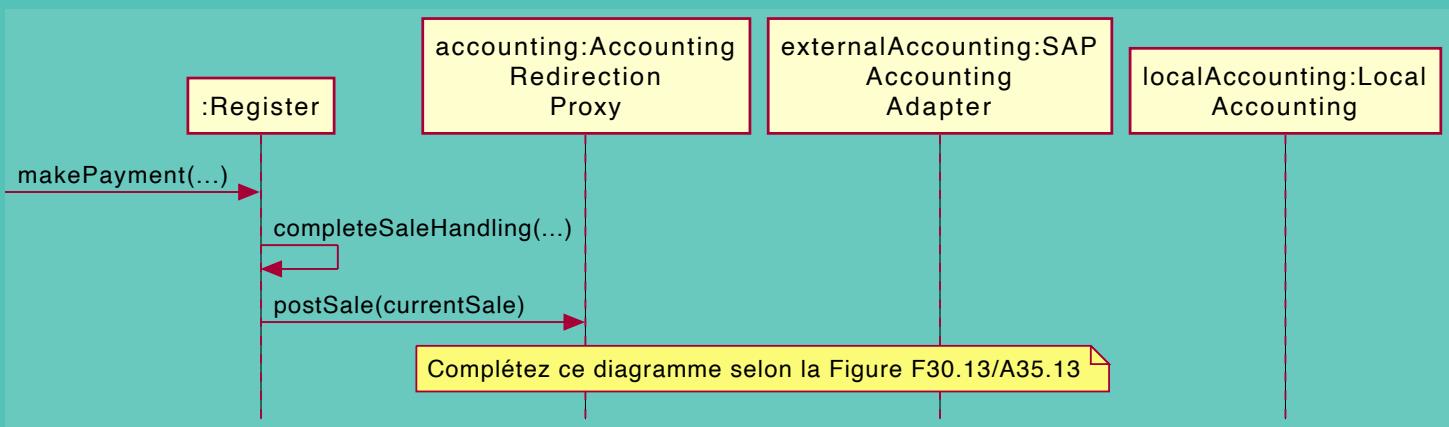


PROXY DE REDIRECTION



Created by Jonathan Li
from the Noun Project

Exercice



ENQUÊTE SUR LA RÉMUNÉRATION D'INGÉNIEUR

- <https://www.genium360.ca/fr-ca/services-aux-entreprises/enquete-sur-la-remuneration/>
- https://drive.google.com/file/d/1UnInAz3y1it-I5S_ED2W-dj2JVJ8cITC/view?usp=sharing

RÉSAUTAGE

[https://drive.google.com/open?
id=1Cm2WTaG5KN0sqonAfOnbxjcuNfbMYFBW](https://drive.google.com/open?id=1Cm2WTaG5KN0sqonAfOnbxjcuNfbMYFBW)

DEVELOPER SURVEY RESULTS

2019

<https://insights.stackoverflow.com/survey/2019>

27 . 24

LISTE DE RECRUTEURS

[https://drive.google.com/open?
id=11XYE6k8m6YIcbAJ6hGBfkmIIJLH_DRTp](https://drive.google.com/open?id=11XYE6k8m6YIcbAJ6hGBfkmIIJLH_DRTp)

[https://drive.google.com/open?
id=1jJT5YChfylxJNfSQQPH1lsBLab2CQYpG](https://drive.google.com/open?id=1jJT5YChfylxJNfSQQPH1lsBLab2CQYpG)

27 . 26

97 THINGS

97-THINGS-EVERY-PROGRAMMER-SHOULD-KNOW

- <https://github.com/97-things/97-things-every-programmer-should-know/blob/master/en/SUMMARY.md>

97-THINGS-EVERY-AGILE-DEVELOPER-SHOULD-KNOW

- <https://github.com/97-things/97-things-every-agile-developer-should-know>

