


# LOG210 SÉANCE #06

## ANALYSE ET CONCEPTION DE LOGICIELS



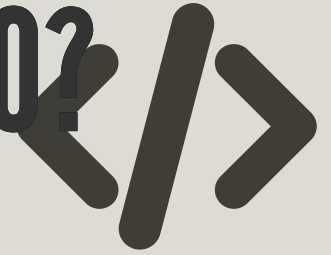
1. **AOO** - Approche orienté objet  **S20203**
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra

# 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

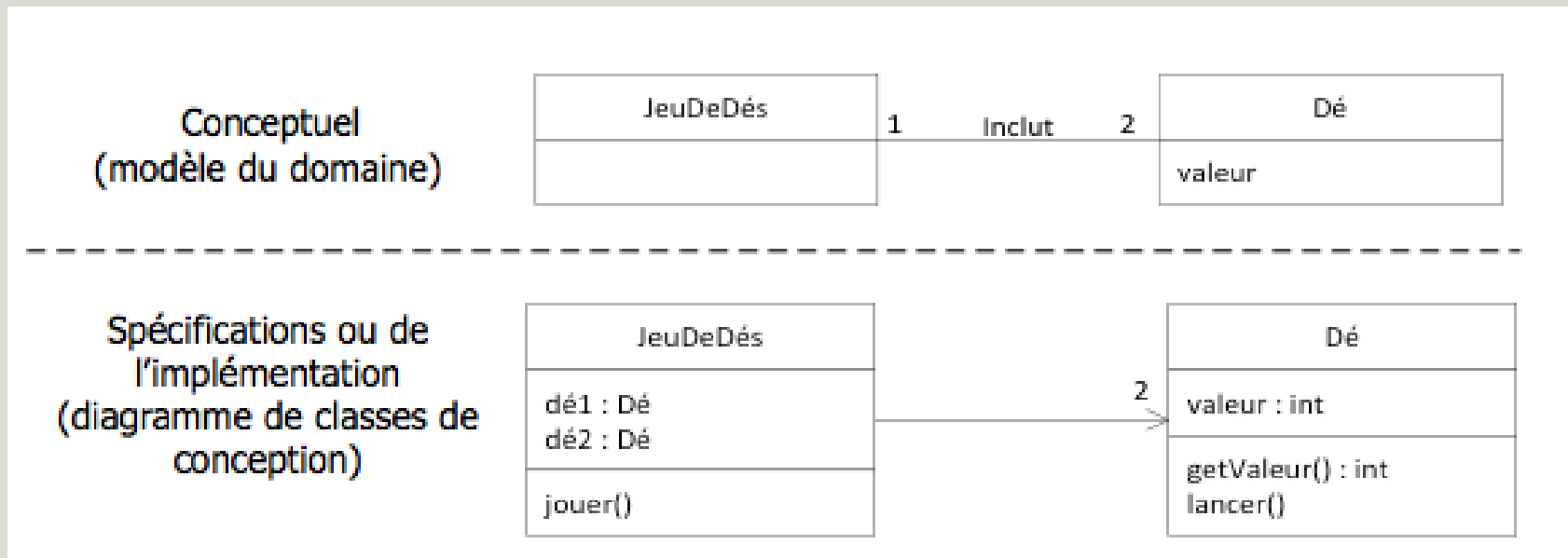


- « Langage visuel dédié à la spécification, la construction et la documentation des artefacts d'un système » – OMG
- Trois façons d'appliquer UML
  - Mode esquisse – diagrammes informels et incomplets, tracés à la main
  - Mode en plan – diagrammes détaillés
  - Comme langage de programmation – spécification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

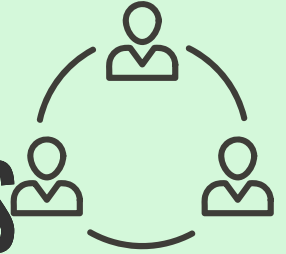
Avion
numéroDeVol


Avion
numéroDeVol
getHistoriqueDuVol()

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

# LOG210 SÉANCE #06

## ANALYSE ET CONCEPTION DE LOGICIELS



1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres  **S20203**
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra



# RENCONTRES EFFICACES

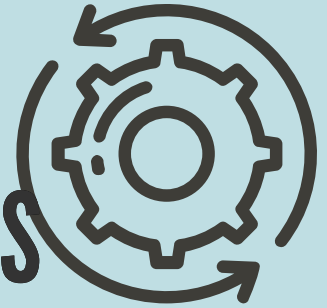


1. N'inviter que ceux qui doivent absolument être présents.
2. Préparer et distribuer l'ordre du jour bien avant que ça commence.
3. Terminer tôt si les objectifs ont été atteints.
4. Respecter l'ordre du jour.
5. Planifier autour des pauses habituelles (p.ex., à midi, à la fin de la journée).

[TeamGeek]

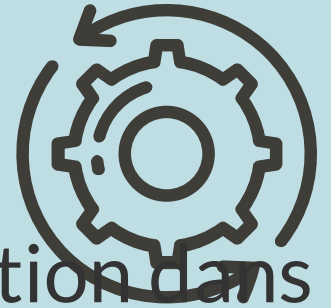
# LOG210 SÉANCE #06

## ANALYSE ET CONCEPTION DE LOGICIELS



1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié ← **S20203**
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra

# PROCESSUS UNIFIÉ?

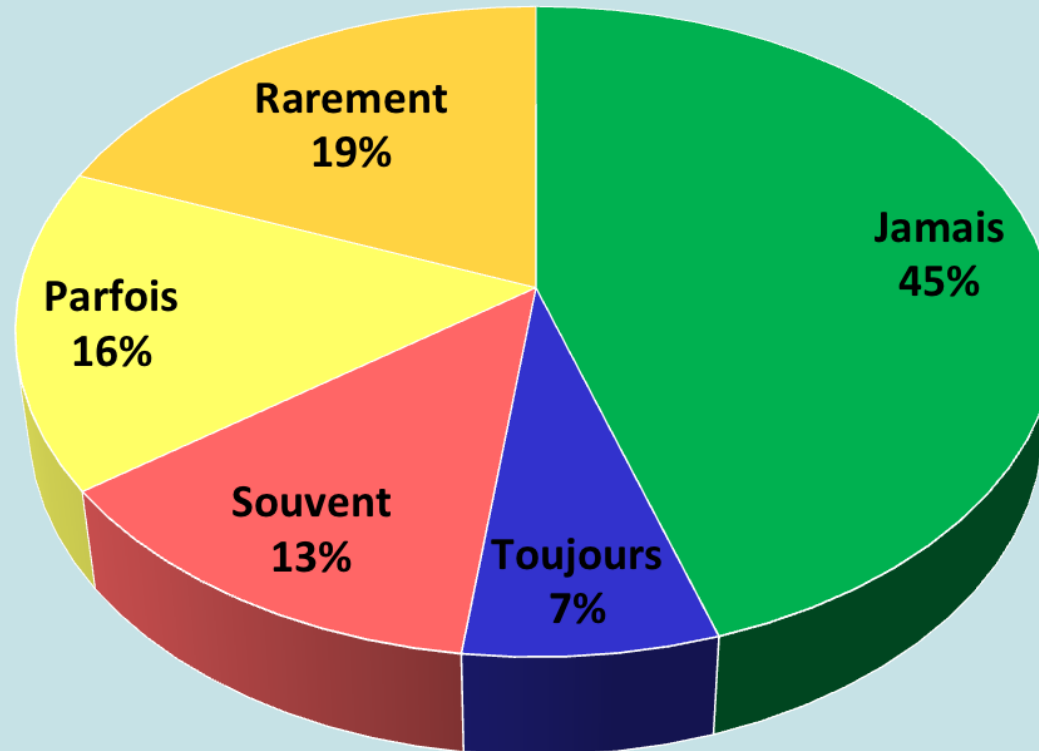


- Quelle est une durée typique d'une itération dans le processus unifié?

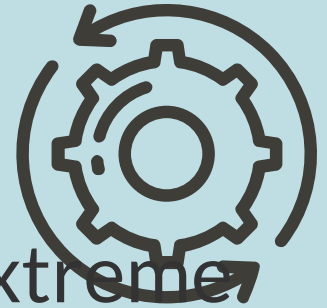
# APPROCHE ÉVOLUTIVE VS APPROCHE EN CASCADE



Utilisation réelle des fonctionnalités spécifiques dans un processus en cascade

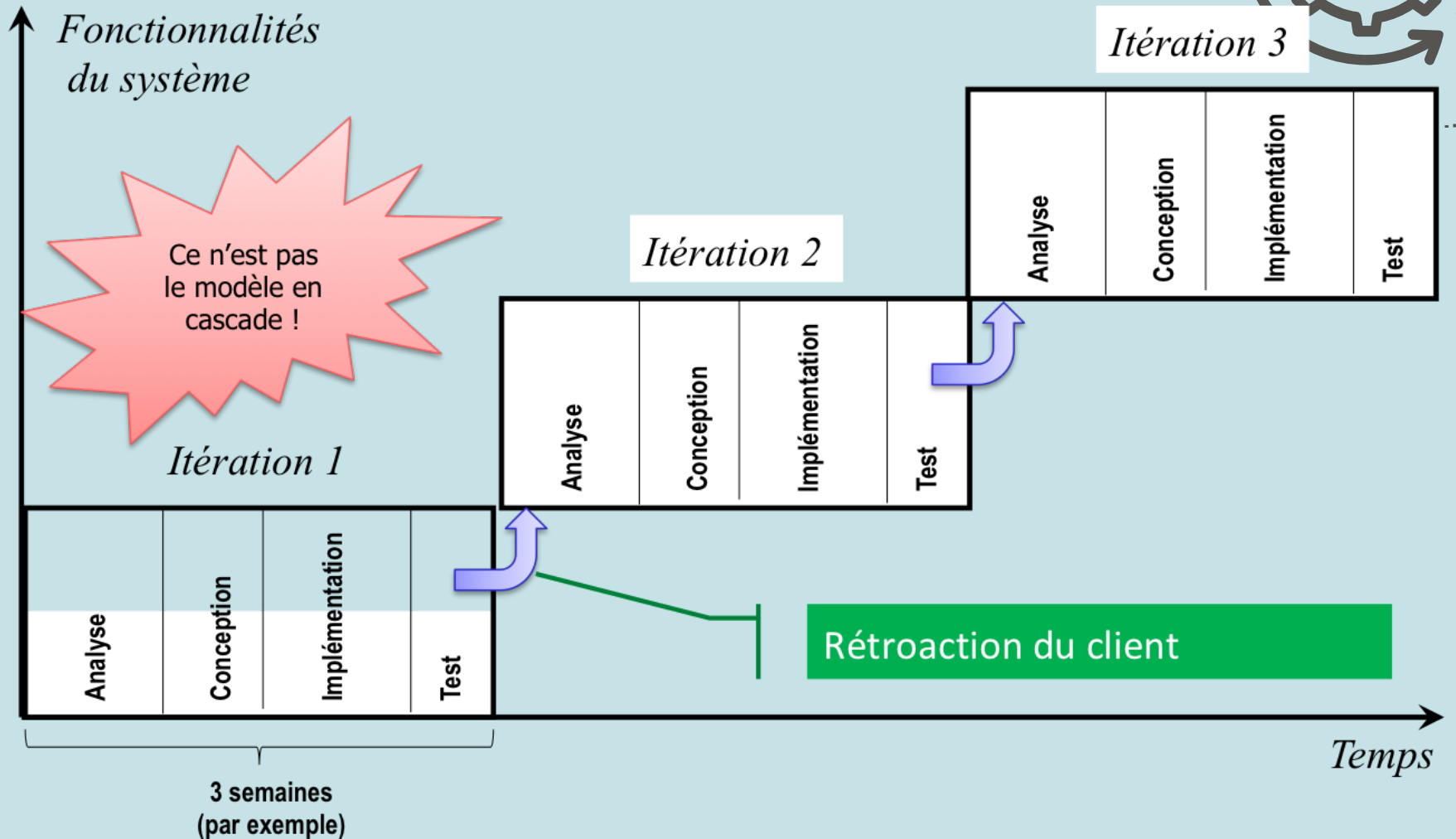
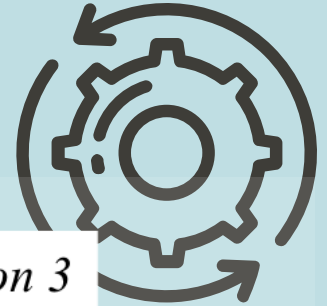


# PROCESSUS UNIFIÉ



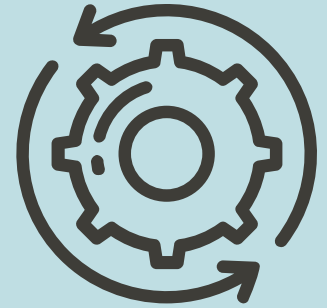
- Souple et ouvert – peut accommoder l'Extreme Programming (XP), Scrum, etc.
- Développement piloté par les tests
  - intégration continue « war room »
  - réunions quotidiennes
  - etc.
- Processus *itératif* et *évolutif*

# ITÉRATIF ET ÉVOLUTIF





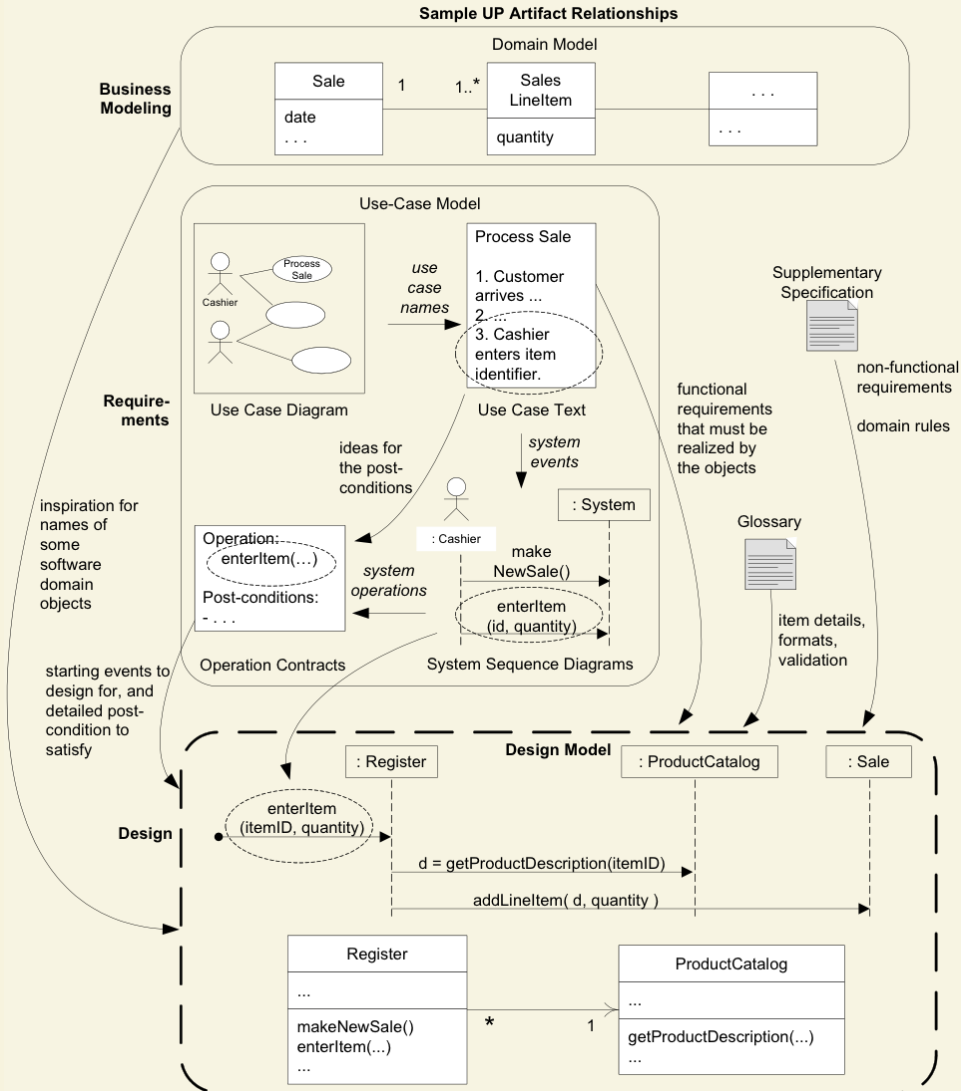
# RÉSUMÉ



- 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)



# RELATION ENTRE ARTEFACTS – CONCEPTION




# LOG210 SÉANCE #06

## ANALYSE ET CONCEPTION DE LOGICIELS



Created by Bharat  
from the Noun Project

1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS  S20203\$
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra

# FURPS - EXIGENCES DU CLIENT



Created by Bharat  
from the Noun Project

- **FURPS+**
  1. **F**onctionnalité (Use case)
  2. **U**sability (Aptitude à l'utilisation, facteurs humains, aide et documentation)
  3. **R**eliability (Fiabilité)
  4. **P**erformance
  5. **S**upportability (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:

- *Façade* 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.

Plus d'exemples.

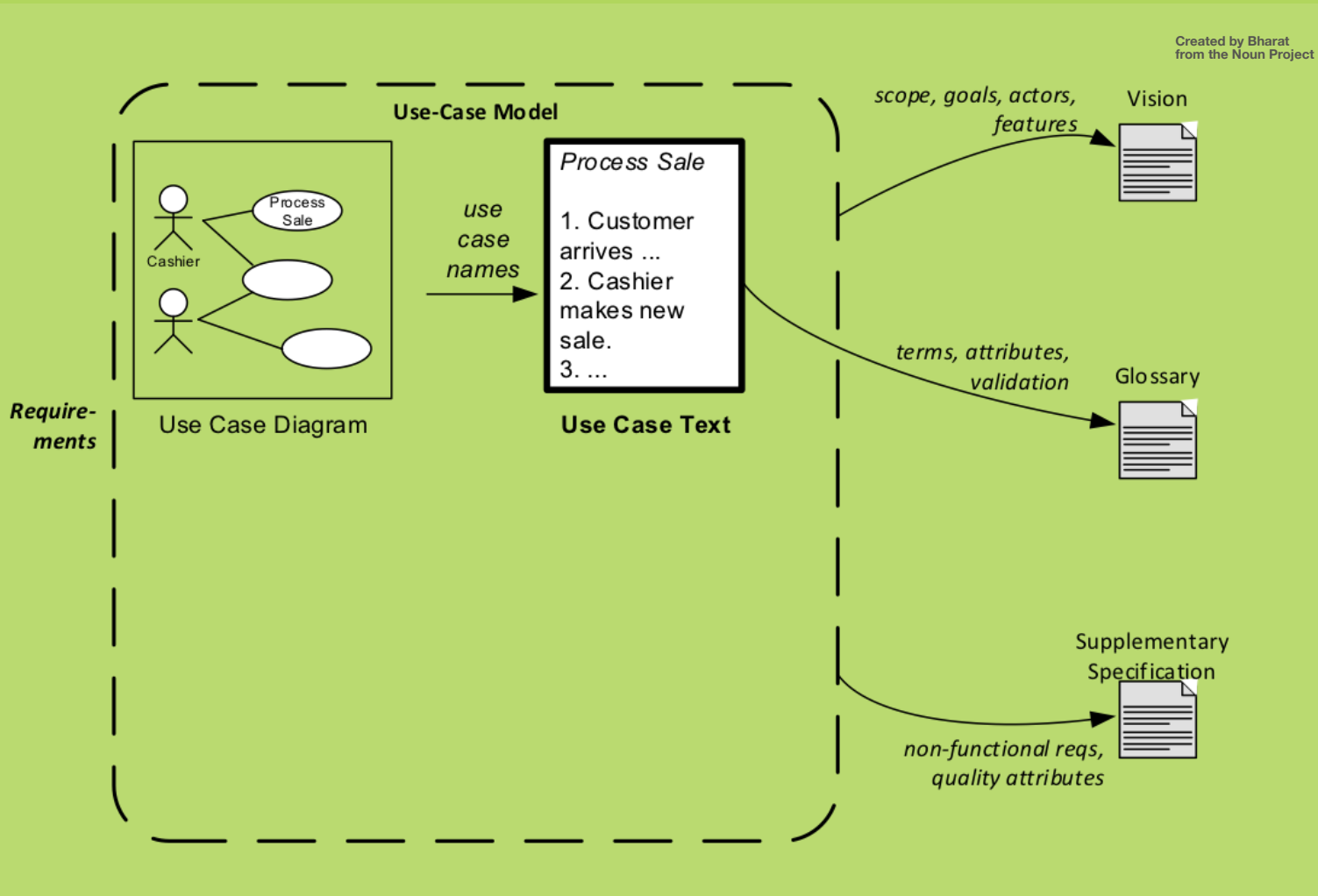


Created by Bharat  
from the Noun Project

# 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




- 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

Created by Bharat  
from the Noun Project

## ANALYSE ET CONCEPTION DE LOGICIELS



Created by Adrien Coquet  
from the Noun Project

1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation 
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra

# 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.

Created by Lucie Thibault  
from the Joun Project

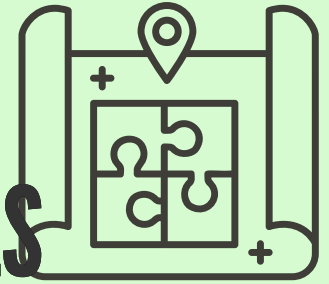
# ACTEUR




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

Created by Adrien Coquet  
from the Noun Project

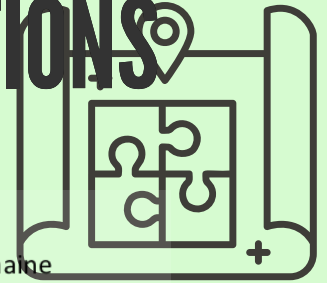




## ANALYSE ET CONCEPTION DE LOGICIELS

1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine  **S20203**
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra

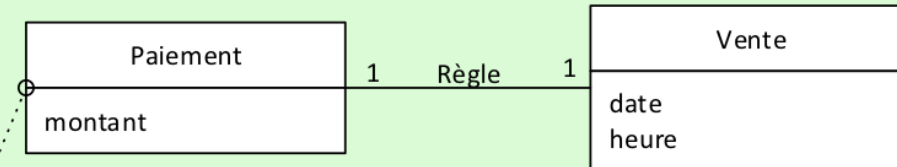
# RÉDUIRE LE DÉCALAGE DES REPRÉSENTATIONS



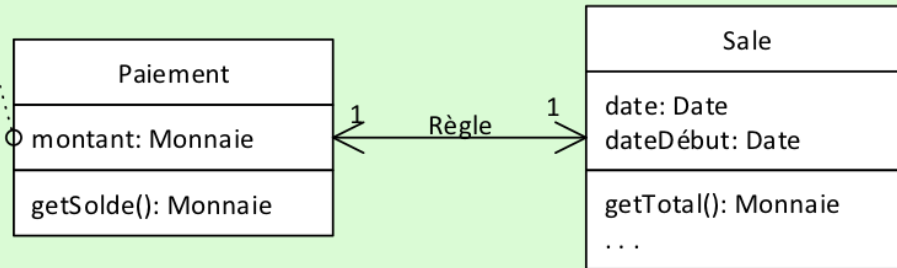
Dans le Modèle du Domaine, un Paiement est un concept, mais c'est une classe logicielle dans le Modèle de Conception. Les deux ne sont pas identiques, et le premier a *inspiré* le nom et la définition de la seconde.

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 les concepts significatifs du domaine



inspire les  
objets et les  
noms dans



**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?



# 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



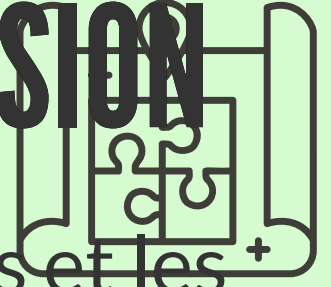
1. Qu'est-ce que la complexité inhérente?
2. Donnez un synonyme de inhérente.
3. Qu'est-ce que la complexité accidentelle?
4. La complexité inhérente est-elle due au problème ou à la solution?
5. Est-il plus facile de gérer la complexité inhérente ou accidentelle? Pourquoi?

# 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?

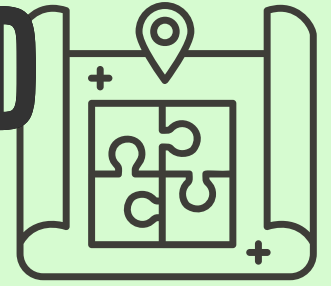
# VALIDATION DE LA COMPRÉHENSION



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

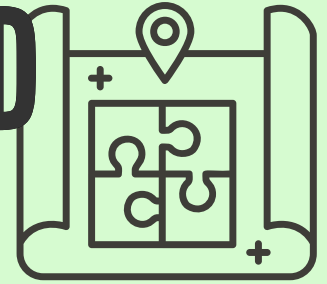
- Vrai
- Faux

# CHP26 AFFINEMENT DU MDD



1. Qu'est-ce que la généralisation?

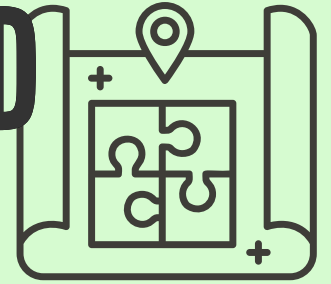
# CHP26 AFFINEMENT DU MDD



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

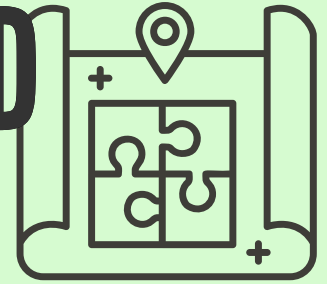


# CHP26 AFFINEMENT DU MDD



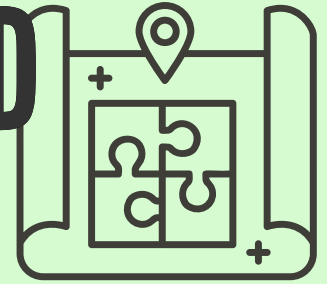
1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?

# CHP26 AFFINEMENT DU MDD



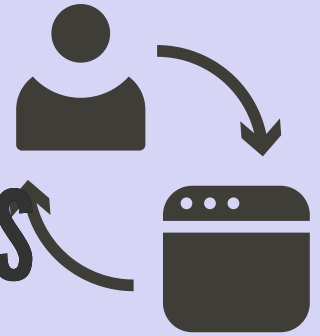
1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?
4. Est-il préférable d'utiliser une agrégation ou une composition? Pourquoi?

# 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?

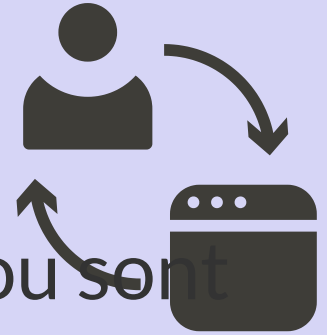
# LOG210 SÉANCE #06



## ANALYSE ET CONCEPTION DE LOGICIELS

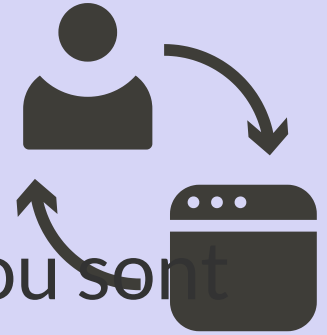
1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système  **S20203\$**
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra

# CHP10 DSS



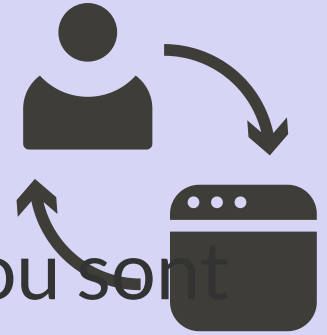
1. Quels sont les artéfacts qui influencent ou sont influencés par le DSS?

# CHP10 DSS



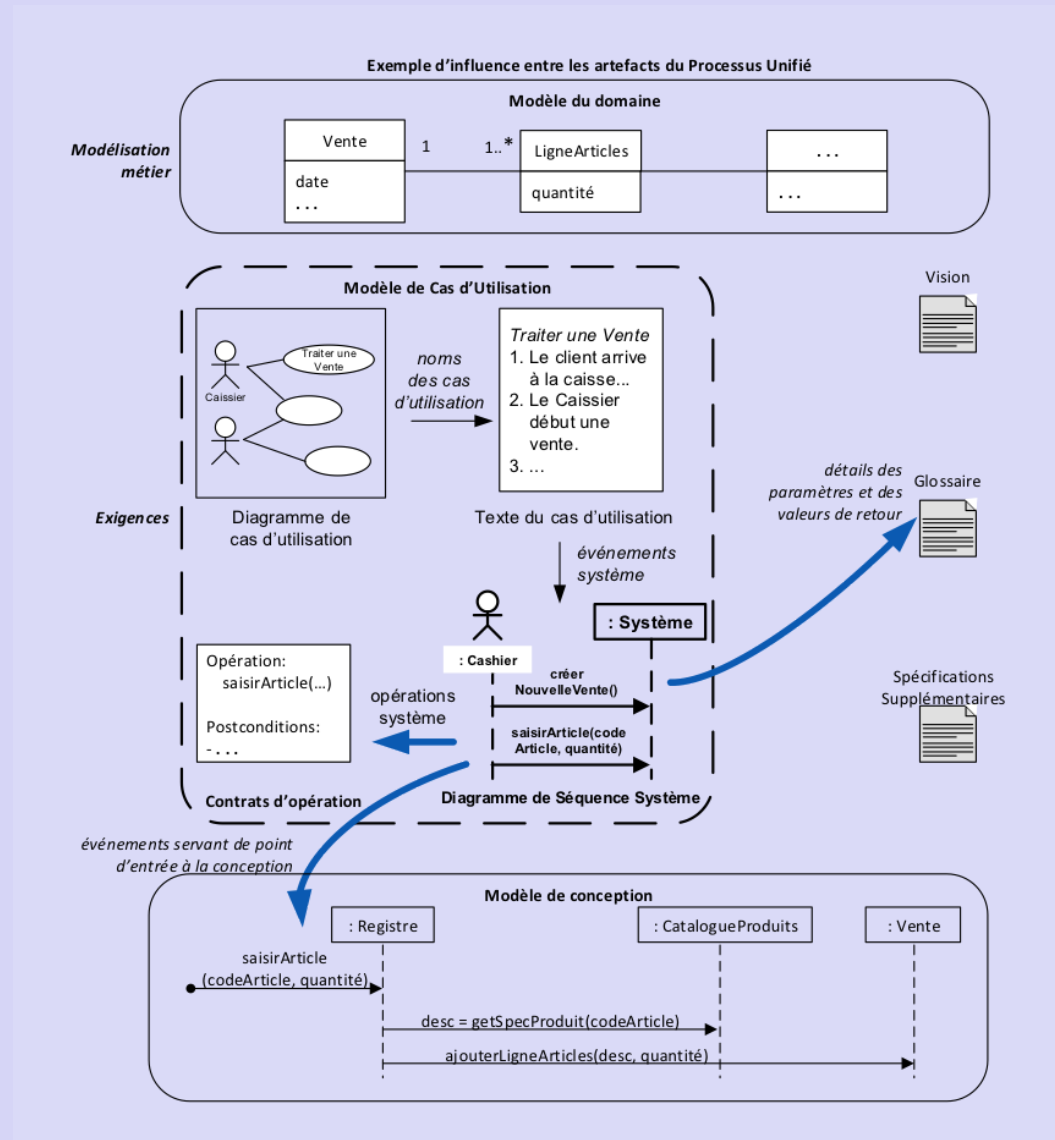
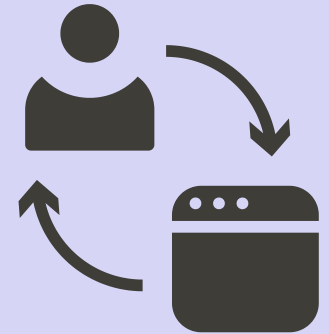
1. Quels sont les artéfacts qui influencent ou sont influencés par le DSS?
2. Qu'est-ce qu'un DSS?

# CHP10 DSS



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

# INFLUENCE ENTRE ARTEFACTS DU PU







# LOG210 SÉANCE #06



## ANALYSE ET CONCEPTION DE LOGICIELS

Created by Mohammed Rai

1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra



# CHP11 CONTRATS

1. Quels sont les éléments d'un contrat?

# CHP11 CONTRATS

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

# CHP11 CONTRATS

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



# CONTRATS - ERREURS FRÉQUENTES



Created by Mohammed Rai

- 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

## ANALYSE ET CONCEPTION DE LOGICIELS



Created by Swen-Peter Ekkebus  
from the Noun Project

1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra





# CHP15 DIAGRAMME DE CLASSE

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

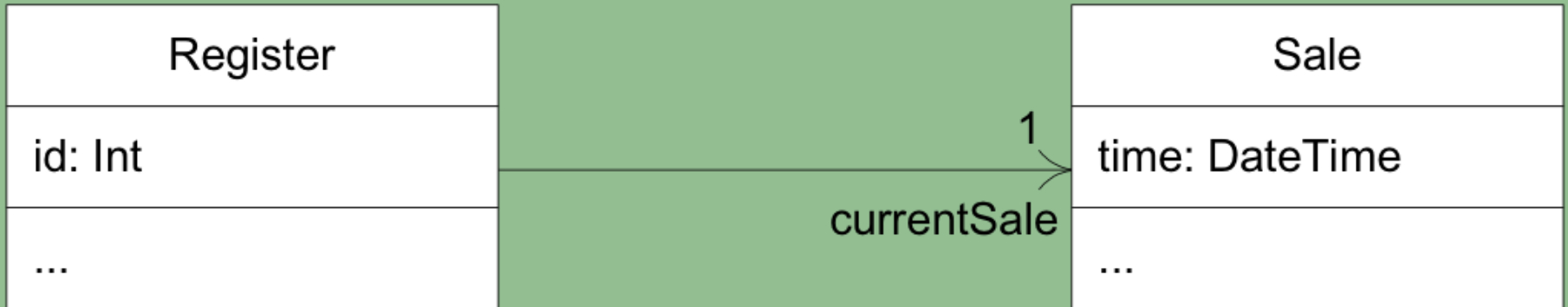
# CHP15 DIAGRAMME DE CLASSE

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

# CHP15 DIAGRAMME DE CLASSE

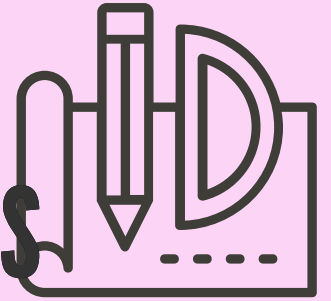
1. Quelle est la différence entre un diagramme de classes conceptuelles et logicielles?
2. Quelles sont les deux façons de noter les attributs?
3. Pourquoi fait-on un DCC?

# VALIDATION DE LA COMPRÉHENSION




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

# LOG210 SÉANCE #06



## ANALYSE ET CONCEPTION DE LOGICIELS

1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation  **S20203**
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra

# CHP17 CONCEPTION AVEC LES GRASP

## 1. Qu'est-ce qu'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?

# CHP17 CONCEPTION AVEC LES GRASP

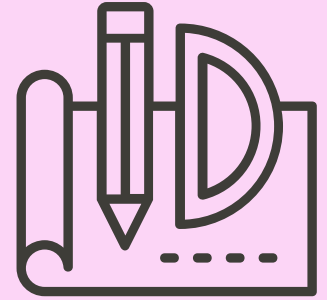
1. Qu'est-ce qu'une RDCU?
2. Quelle est l'utilité des contrats d'opérations dans une RDCU?
3. Est-ce que le retour d'information dans une opération du DSS est important pour la création d'une RDCU?



# CHP17 CONCEPTION AVEC LES GRASP

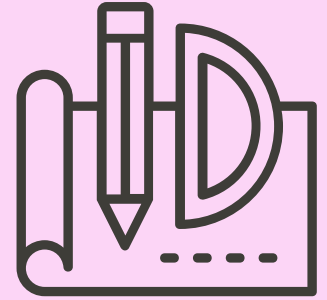
1. Qu'est-ce qu'une RDCU?
2. Quelle est l'utilité des contrats d'opérations dans une RDCU?
3. Est-ce que le retour d'information dans une opération du DSS est important pour la création d'une RDCU?
  - Pourquoi?

# CHP18 VISIBILITÉ



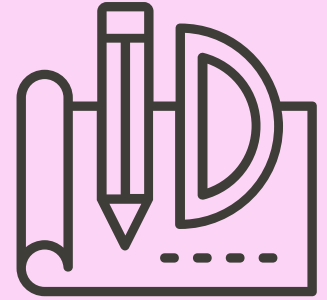
1. Nommez les 4 types de visibilité.

# CHP18 VISIBILITÉ



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

# CHP18 VISIBILITÉ



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

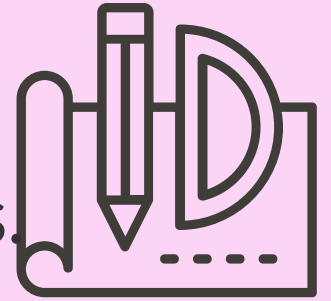
# EXERCICE RDCU

Correction

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


# RDCU - AIDE MÉMOIRE

Voir la figure 8.1 des notes de cours.



## ANALYSE ET CONCEPTION DE LOGICIELS



1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test  **S20203** Problème étudiant
12. **Intra** - modalités intra

# 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.

# 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.

# 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.

# LOG210 SÉANCE #06

## ANALYSE ET CONCEPTION DE LOGICIELS



Created by Vectors Point

1. **AOO** - Approche orienté objet
2. **Equipe** - Travail d'équipe - Rencontres
3. **PU** - Phases du Processus Unifié
4. **FURPS** - FURPS
5. **CU** - Cas d'utilisation
6. **MDD** - Modèle du domaine
7. **DSS** - Diagramme de séquence système
8. **Contrats** - Contrats pour réaliser les RDCU
9. **DCL** - Diagramme de classes de logicielles
10. **RDCU** - Réalisation d'un cas d'utilisation
11. **TDD** - Développement piloté par les test
12. **Intra** - modalités intra



# EXAMEN INTRA



- Toute la matière du cours et du laboratoire
- Exercices sur Google Classroom

Created by Vectors Point



# DÉROULEMENT 1/2



- Examen sur Moodle
  - Examen de pratique - numériser des dessins de diagrammes
  - Toute documentation permise (!)
  - Préparez feuilles de notes

Credited by Vectors Point

# DÉROULEMENT 2/2

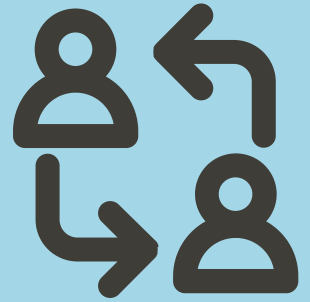


Created by Vectors Point

- Lien Zoom différent (annoncé sur Google Classrooms)
  - Salles Zoom de  $\leq 18$  personnes
  - Surveillants
  - Activez votre webcam
  - Vous connecter sur Zoom avec votre courriel de l'ÉTS @ens.etsmt1.ca pour faciliter les choses

# SÉANCE #06

## RÉTROACTION: PAGE D'UNE MINUTE



Created by Prithvi  
from the Noun Project

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

<https://1drv.ms/u/s!An6-F73ulxAOhVyiCB46jTeINVLs>

