

LOG210 ANALYSE ET CONCEPTION DE LOGICIELS: SÉANCE 05

SURVOL

- Travail en équipe
- Rétroaction mini-test
- Retour Exercices
- TDD
- RDCU: Faible couplage/Forte Cohésion



CULTURE D'ÉQUIPE

DÉVELOPPEMENT DE LOGICIEL

CULTURE D'ÉQUIPE





QU'EST-CE QU'UNE CULTURE D'ÉQUIPE?

- Ensemble de valeurs, d'objectifs, d'expériences
- Unique à chaque équipe



ÉLÉMENTS TECHNIQUES

- Revues de code
- Développement piloté par les tests
- Documentation de la conception
- etc.



ÉLÉMENTS SOCIAUX

- Sushi à midi
- 5 à 7 le vendredi
- etc.



QU'EST-CE QU'UNE CULTURE D'ÉQUIPE?

- Qu'elle soit bonne ou mauvaise, la culture existera
- Le/La leader ne décide pas la culture; il/elle s'en occupe.



CULTURE FORTE D'ÉQUIPE

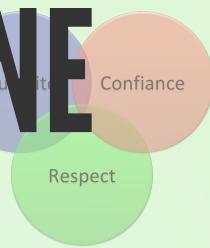
- Ouverte au changement qui l'améliore
- Résistante à un changement radical qui lui fait mal



CULTURE FORTE D'ÉQUIPE

- 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

UNE CULTURE S'AUTO-SÉLECTIONNE



Ne faire que du hacking cowboy,
initiation (bizutage), dérapages
verbaux, agressivité



Écrire des tests, réviser le design
et le code, valoriser l'inclusivité



ÉLÉMENTS POUR LES CULTURES D'ÉQUIPE RÉUSSITES

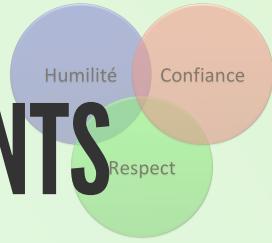
- Énoncé de mission d'équipe
- Culture de HRC
- Favorise une participation des coéquipiers extrovertis et introvertis
- team.égo > coéquipier[i].égo
- La critique constructive
- Bonne communication
- Moyens de communication diversifiés (Slack, courriel, etc.)
- Documentation de design
- Communication 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.

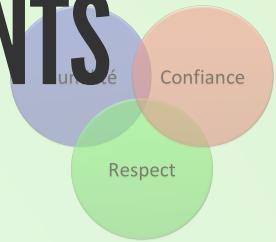


COMMENT AGIR FACE À CES COMPORTEMENTS

- 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...



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).

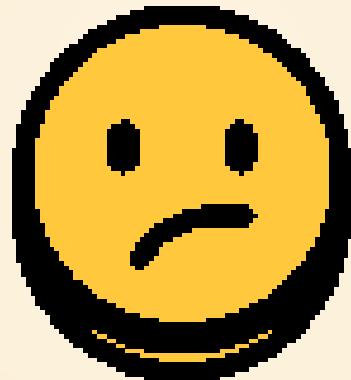


RÉTROACTION

MINI-TEST



QUESTIONS DIFFICILES



Selon les statistiques de hier matin.



Socrative → ETSLOG210

Quel est le sens de **facteur de bus** dans le travail en équipe?

- A. nombre de coéquipiers travaillant sur les tâches différentes
- B. degré de redondance des compétences
- C. probabilité de perdre un coéquipier



Socrative → ETSLOG210

Quelle est la valeur de retour de

```
typeof (new Voiture())
```

- A. Voiture
- B. "Voiture"
- C. object
- D. any



Socrative → ETSLOG210

TS

Qu'affiche le programme suivant?

```
1 let maMap = new Map<number, string>() ;  
2 maMap.set(77, 'Poisson') ;  
3 maMap.set(22, 'Citron') ;  
4 console.log(maMap.get(77)) ;  
5 console.log(maMap.has(22)) ;
```

- A. true / Citron
- B. Poisson / Citron
- C. Poisson / undefined
- D. Poisson / true



Socrative → ETSLOG210

TS

Qu'affiche le programme suivant?

```
1 let maMap = new Map<number, string>() ;  
2 maMap.set(77, 'Poisson') ;  
3 maMap.set(22, 'Citron') ;  
4 console.log(maMap.get(77)) ;  
5 console.log(maMap.has(22)) ;
```

- A. true / Citron
- B. Poisson / Citron
- C. Poisson / undefined
- D. Poisson / true



Socrative → ETSLOG210

TS

Qu'affiche le programme suivant?

```
1 let maMap = new Map<number, string>() ;  
2 maMap.set(77, 'Poisson') ;  
3 maMap.set(22, 'Citron') ;  
4 console.log(maMap.get(77)) ;  
5 console.log(maMap.has(22)) ;
```

- A. true / Citron
- B. Poisson / Citron
- C. Poisson / undefined
- D. Poisson / true



Socrative → ETSLOG210

TS

Qu'affiche le programme suivant?

```
1 let maMap = new Map<number, string>() ;  
2 maMap.set(77, 'Poisson') ;  
3 maMap.set(22, 'Citron') ;  
4 console.log(maMap.get(77)) ;  
5 console.log(maMap.has(22)) ;
```

- A. true / Citron
- B. Poisson / Citron
- C. Poisson / undefined
- D. Poisson / true



Socrative → ETSLOG210

TS

Qu'affiche le programme suivant?

```
1 let maMap = new Map<number, string>() ;  
2 maMap.set(77, 'Poisson') ;  
3 maMap.set(22, 'Citron') ;  
4 console.log(maMap.get(77)) ;  
5 console.log(maMap.has(22)) ;
```

- A. true / Citron
- B. Poisson / Citron
- C. Poisson / undefined
- D. Poisson / true

Tableau associatif et get vs has



Socrative → ETSLOG210

TS

Complétez le programme suivant pour qu'il produise le message [67, 46, 14, 6]:

```
const valeurs = [ 65, 44, 12, 4 ];  
console.log(valeurs.map( _____ ) );
```

- A. $v \Rightarrow v + 2$
- B. $v + 2$
- C. $v \Rightarrow v - 2$
- D. $v - 2$



Socrative → ETSLOG210

TS

Complétez le programme suivant pour qu'il produise le message [67, 46, 14, 6]:

```
const valeurs = [65, 44, 12, 4];
console.log(valeurs.map( _____ ) );
```

- A. $v \Rightarrow v + 2$
- B. $v + 2$
- C. $v \Rightarrow v - 2$
- D. $v - 2$

Voir cet exemple.



Socrative → ETSLOG210

Quel principe GRASP a l'objectif principal de minimiser les dépendances dans la conception?

- A. Forte Cohésion
- B. Faible Couplage
- C. Expert en information
- D. Contrôleur



Socrative → ETSLOG210

Selon le principe **Créateur**, vous trouvez que deux classes différentes pourraient avoir la même responsabilité de créer une instance (objet) d'une classe. Quel(s) principe(s) GRASP existe(nt) pour vous aider à décider laquelle est la meilleure des classes pour avoir cette responsabilité?

- A. Expert en information
- B. Faible Couplage
- C. Forte Cohésion
- D. Contrôleur



TDD

Test-driven development

Développement piloté par les tests

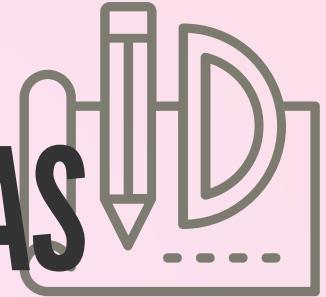
(Notes de cours)

EXAMPLE KATA FIZZBUZZ

Démonstration de Kata FizzBuzz avec JEST (TypeScript) en VSCode



RDCU - RÉALISATION D'UN CAS



D'UTILISATION

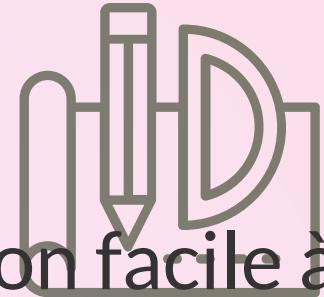
Approche: conception orientée-responsabilités

GRASP

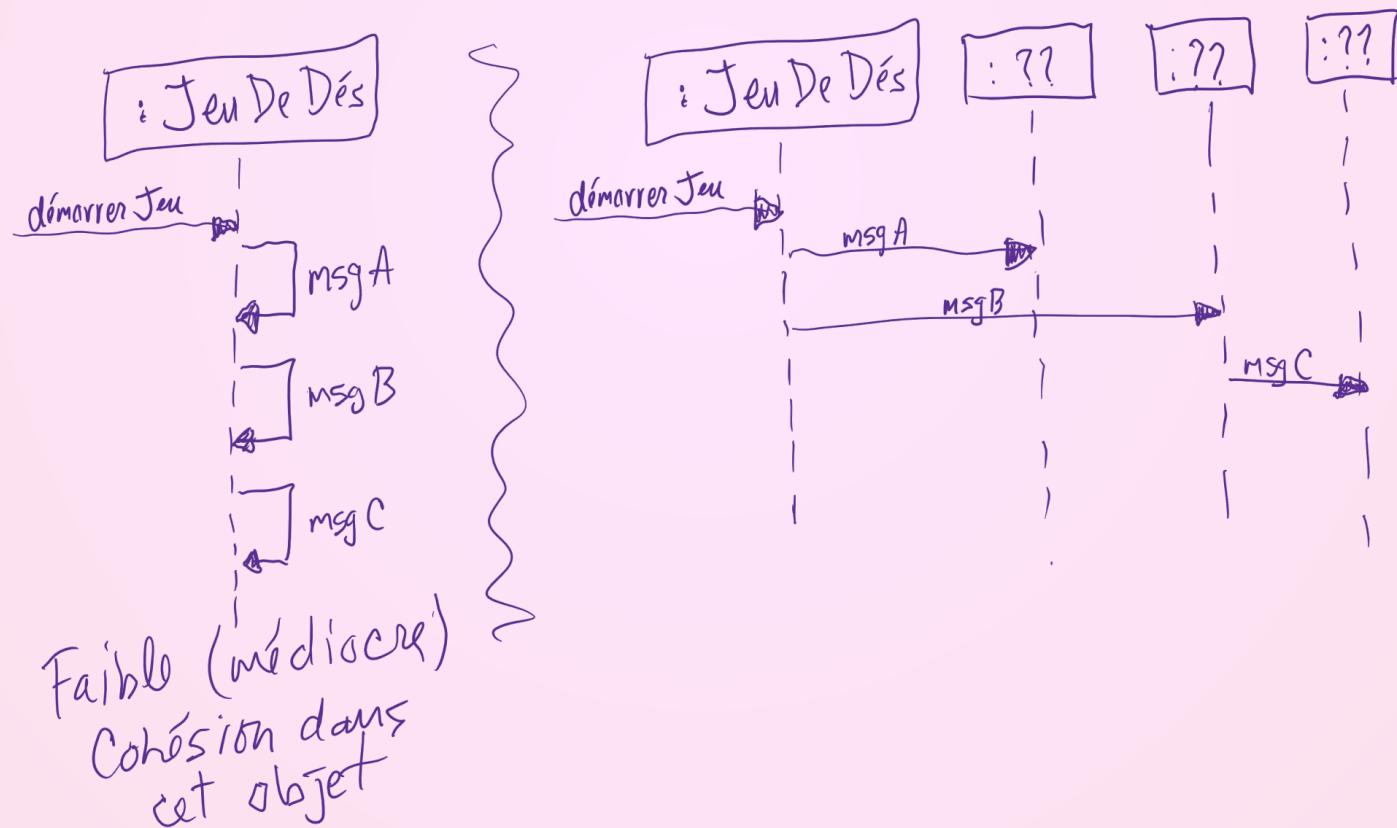
General Responsibility Assignment Software Patterns

Pour décider où mettre les méthodes...

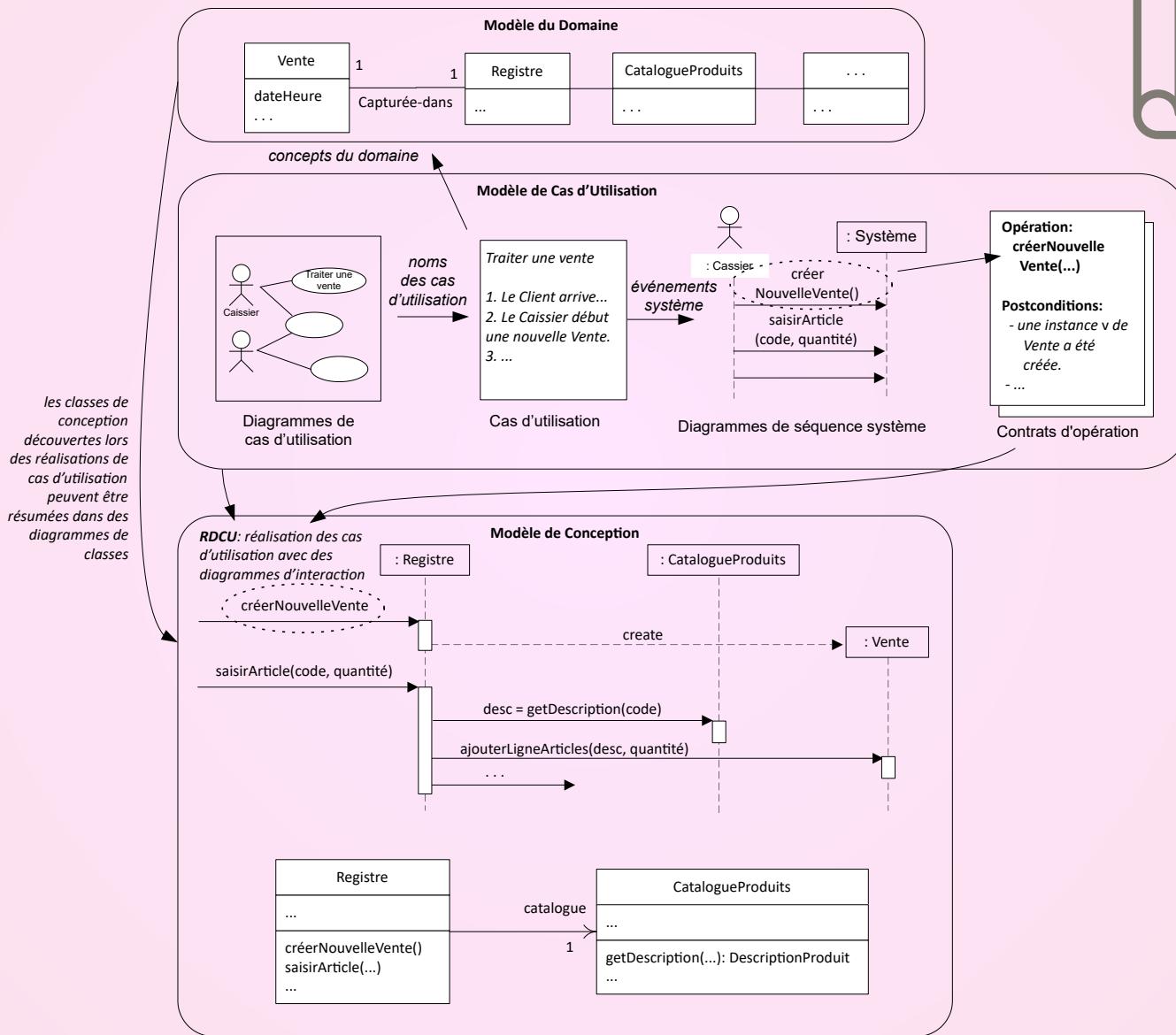
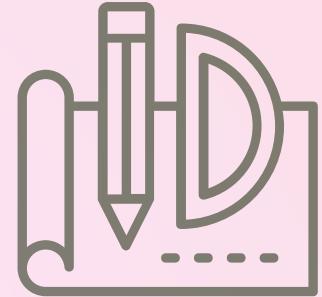
RDCU



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



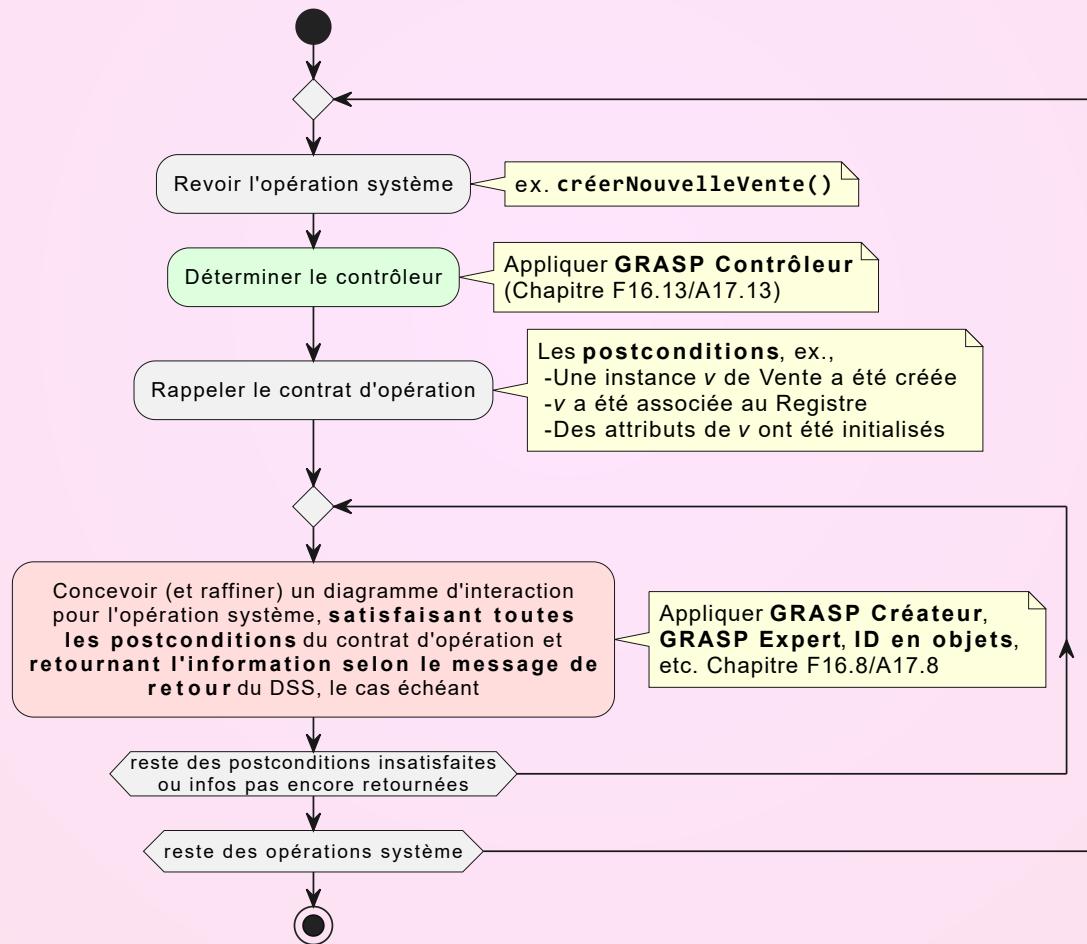
RDCU (SURVOL)

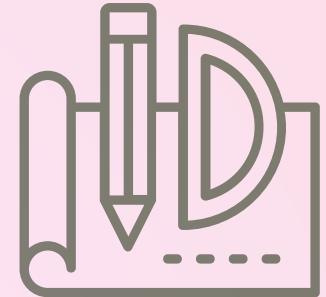


AIDE MÉMOIRE POUR RDCU (FIG. 9.1 NDC)



Note: Une solution détaillée est faite pour chaque opération système.
Donc, il faut utiliser le DSS, les contrats d'opération, le MDD et les principes GRASP pour ce travail.



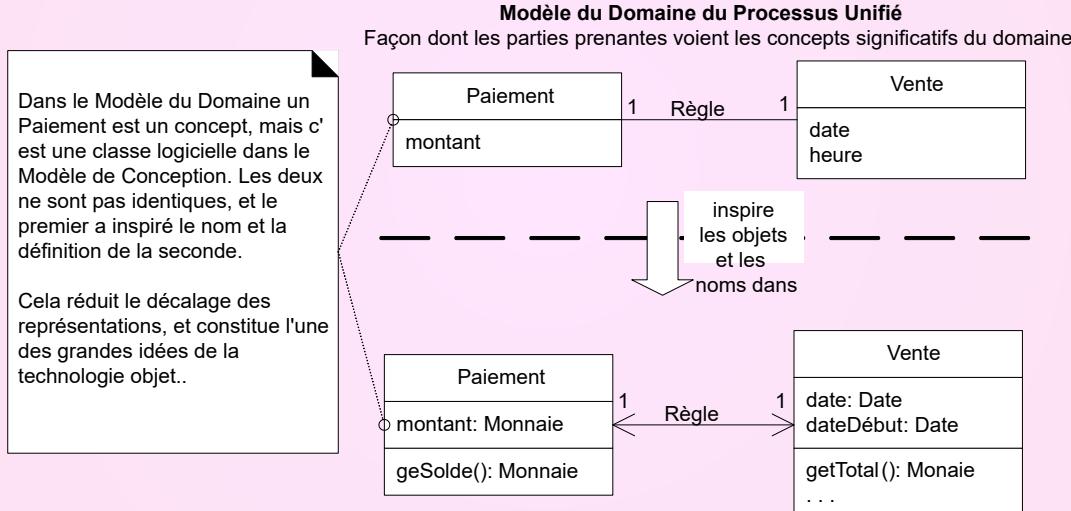


RETOUR TP 04

EXERCICE EN ÉQUIPE

DÉCALAGE DES REPRÉSENTATIONS

Facile? Les classes logicielles devraient ressembler à des classes conceptuelles (concepts du monde réel).

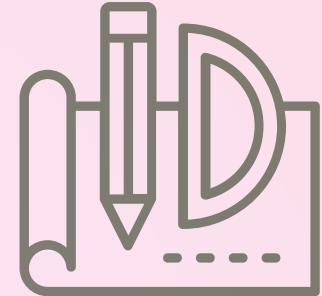


Modèle de Conception du Processus Unifié
Le développeur orienté objet s'est inspiré du domaine du monde réel pour créer les classes logicielles.

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

Qui fait quoi? Qui a quelle responsabilité?

GRASP



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

« ÉVALUATION »

- Faible couplage (évaluation)
- Forte cohésion (évaluation)

On applique ces principes après un autre principe, pour évaluer lorsqu'il y a plusieurs choix de conception.

FAIBLE COUPLAGE

Problème: Comment réduire l'impact des modifications?

Solution: Affecter les responsabilités de sorte à éviter tout couplage inutile. Appliquer ce principe pour évaluer les solutions possibles.

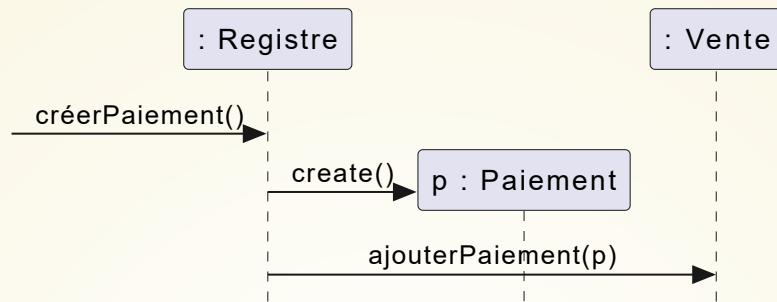
QUI CRÉE UN PAIEMENT?

Selon GRASP Créeateur:

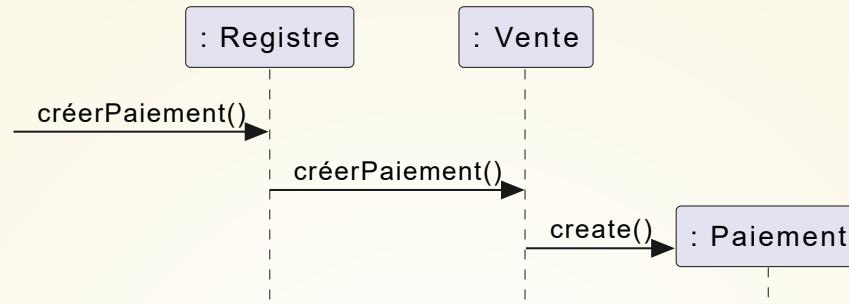
1. un **Registre** « enregistre » un **Paiement**, alors
Registre crée
2. une **Vente** « utilise étroitement » un **Paiement**, alors
Vente crée

Comment décider? On évalue les deux possibilités sur le plan du couplage.

Le Registre crée un Paiement



La Vente crée un Paiement



LE VRAI PROBLÈME DU COUPLAGE

Ce n'est pas le couplage fort en tant que tel. C'est le **couplage fort vers les choses instables**:

- une classe dont son API risque de changer
- les classes de la couche de présentation (vues, le patron Observateur résout ce problème)
- un module externe hors de notre contrôle (dépendances npm sont un bel exemple)

FORTE COHÉSION

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

Solution: Affecter les responsabilités pour que la cohésion demeure élevée. Appliquer ce principe pour évaluer les solutions possibles.

FORTE COHÉSION



Socrative → ETSLOG210

Vrai ou Faux?

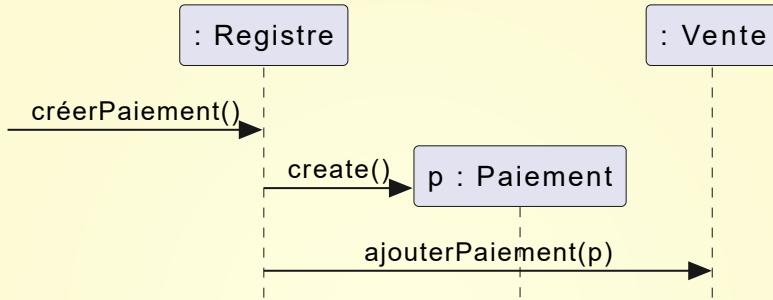


a plus de cohésion que

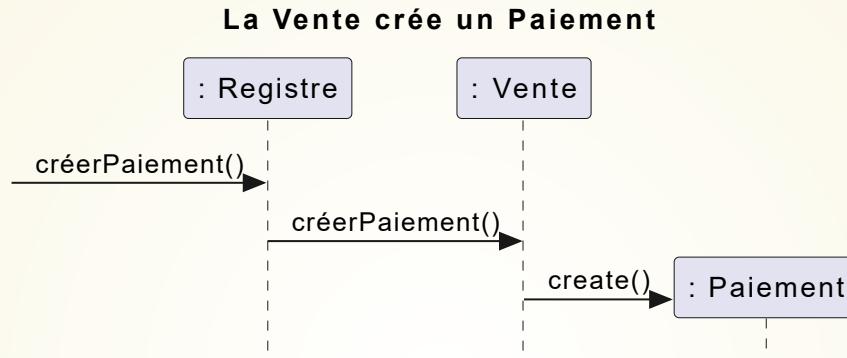


CONSIDÉRER LA COHÉSION

Le Registre crée un Paiement



CONSIDÉRER LA COHÉSION



RÉSUMÉ GRASP

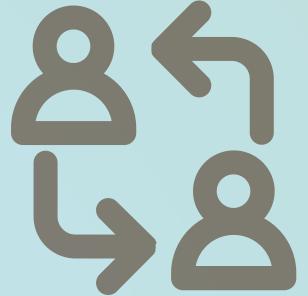
- Faible couplage (évaluation)
- Forte cohésion (évaluation)

EXAMEN INTRA

- Date: vendredi 14 octobre, 2022
- Horaire: 8h30
- Format: Moodle Quiz
- 2h dans un laboratoire informatique:
 - Laboratoire A-3342 : noms A (Abou-Nahed) à G (Gravel)
 - Laboratoire A-3446 : noms H (Hardy) à Z (Zea)
 - ESH Salle B-4404:
 - instructions plus détaillées par courriel
- Aucune documentation n'est permise
- Arriver 15 minutes avant l'examen

EXAMEN INTRA RÉCAPITULATIF

- Méthodologie (artefacts et dépendances)
 - Analyse vs Conception
- Complexités (inhérente, circonstancielle, environnementale)
- FURPS (Functionality, Usability, Reliability, Performance, Supportability)
- Modèle du domaine
- Diagramme de séquence système
- Contrats
- GRASP: Créateur, Expert, Contrôleur, Faible couplage, Forte cohésion
- Spectre de la conception
- Réalisations de cas d'utilisation (RDCU)
 - Scénario “démarrer”



Created by Prithvi
from the Noun Project

FEUILLE D'UNE MINUTE

SVP m'écrire un courriel pour dire ce qu'étaient les points les moins clairs de la séance.