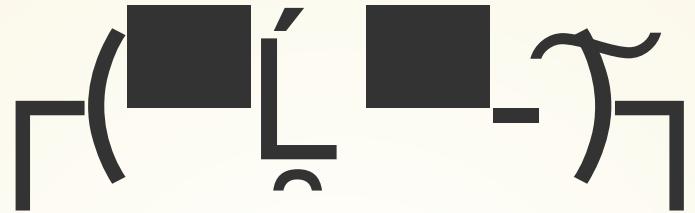


LOG210 ANALYSE ET CONCEPTION DE LOGICIELS: SÉANCE 04



1. Invitation Lab 1 (GitHub Classroom)

- Choisir (ou créer) son équipe
- Suivre les directives du courriel

2. Équipes (envoyées par courriel)

- Équipe peut enlever le nom d'un coéquipier qui n'a rien fait pendant une itération (Lab0)

SURVOL

- Travail en équipe
- Rappel méthodologie
- Rétroaction mini-test
- Retour Exercice TP#3
- GRASP Créateur, Contrôleur et Expert
- RDCU - démarrer



HUMILITÉ

HUMILITÉ



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





HUMILITÉ



HUMILITÉ

- Je ne suis ni omniscient ni infaillible





HUMILITÉ

- Je ne suis ni omniscient ni infaillible



- Je suis ouvert à m'améliorer



HUMILITÉ

- Je ne suis ni omniscient ni infaillible



- Je suis ouvert à m'améliorer
- Moi < Équipe



HUMILITÉ EN PRATIQUE 1/3

- Si quelqu'un travaille sans assez d'humilité
« Veux-tu descendre de ton piédestal? »
- Même si une personne est la plus forte en JavaScript dans l'équipe, si elle le met en évidence constamment, c'est agaçant. 😞
- Dans certaines cultures, l'humilité est très importante (p. ex. le confucianisme). 😊

HUMILITÉ EN PRATIQUE 2/3

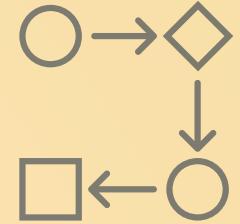


- Apprendre à donner et à accepter les critiques.
 - Donner avec Respect:
« **Ton code est mal écrit.** » vs
« **J'ai de la difficulté à comprendre le flot de contrôle ici dans ton code.** »
 - Accepter avec Humilité:
« **Je comprends ton point de vue. Je vais refactoriser ce code en fin de semaine.** »
- Savoir que son estime de soi n'équivaut pas à sa qualité de code.

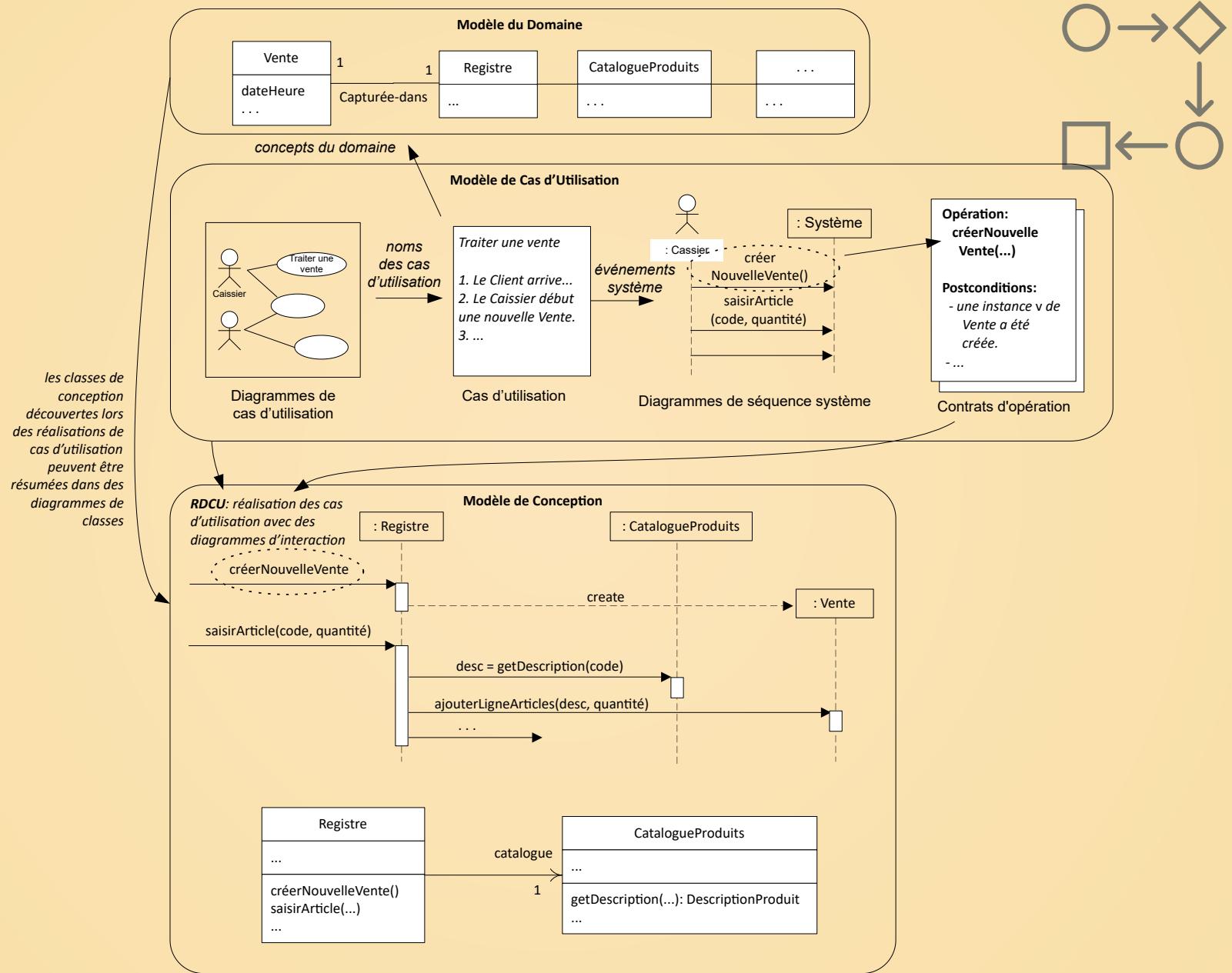


HUMILITÉ EN PRATIQUE 3/3

- 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



MÉTHODOLOGIE

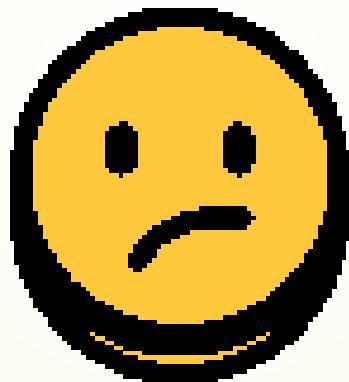


RÉTROACTION

MINI-TEST



QUESTIONS DIFFICILES



Selon les statistiques de hier matin.



Socrative → ETSLOG210

(Question adaptée)

Quelles postconditions sont valides?

Opération:
créerNouvelle
Vente(...)

Postconditions:
- une instance v de
Vente a été
créée.
- ...

- A. Une instance r de Réservation a été créée.
- B. Le Système crée une nouvelle Réservation.
- C. r a été sauvegardée.
- D. $r.date$ est devenue la date actuelle.
- E. r a été associée au Client en cours.
- F. Le Système associe r au Client en cours.



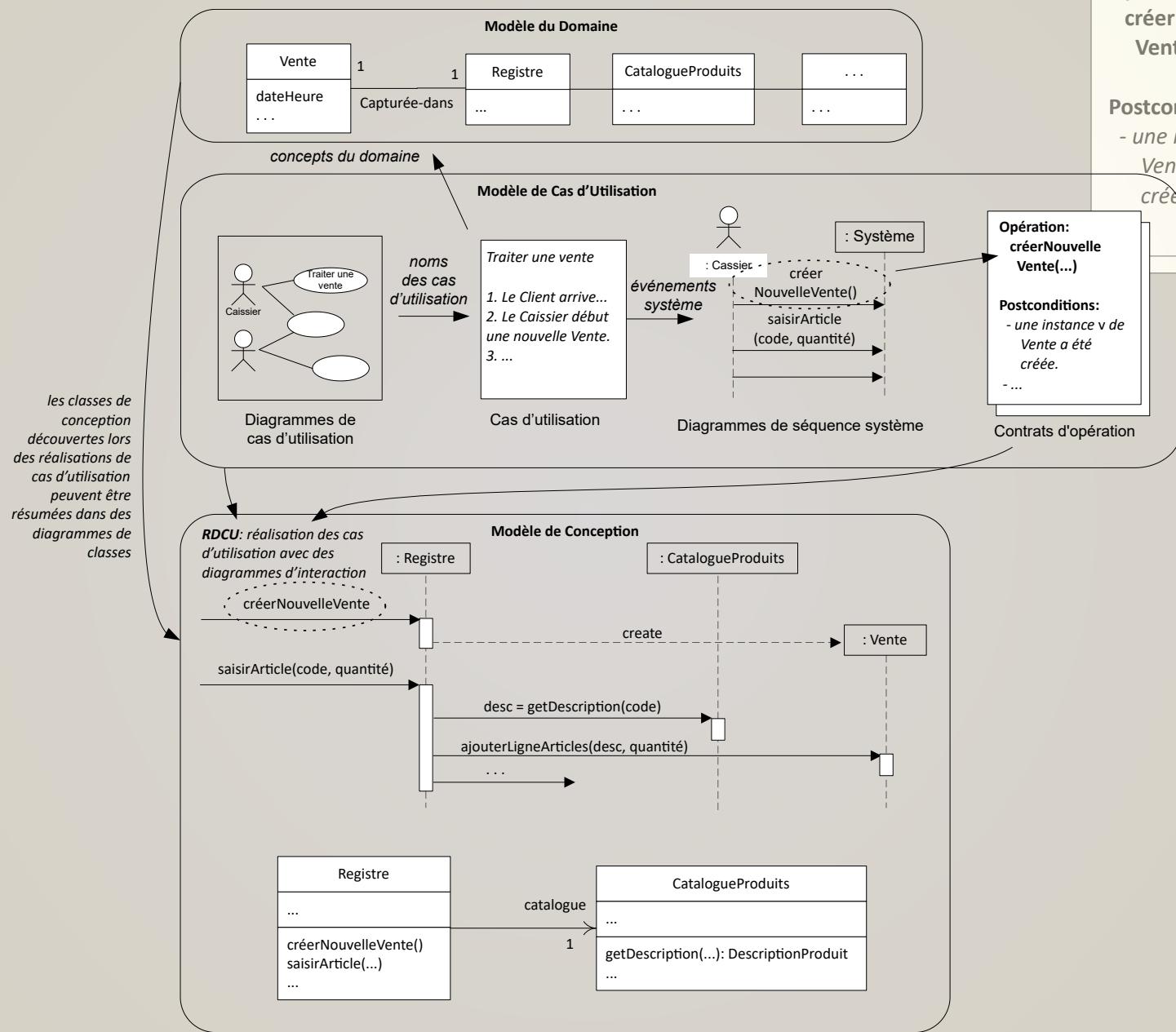
Socrative → ETSLOG210

Quels artefacts du processus unifié sont directement liés aux contrats d'opération?

- A. Cas d'utilisation
- B. Diagramme de séquence système (DSS)
- C. Plan d'itération
- D. Modèle du domaine

Opération:
créerNouvelle
Vente(...)

Postconditions:
- une instance v de
Vente a été
créée.
- ...



Opération:
créerNouvelleVente(...)

Postconditions:

- une instance v de Vente a été créée.

Opération:
créerNouvelleVente(...)

Postconditions:

- une instance v de Vente a été créée.
- ...



Socrative → ETSLOG210

Quel GRASP est toujours appliqué lorsqu'on fait une réalisation de cas d'utilisation?

- A. Fabrication pure
- B. Protection des variations
- C. Contrôleur
- D. Polymorphisme



Socrative → ETSLOG210

Vous décidez de mettre la méthode `getPrix()` dans la classe `DescriptionArticle`, qui possède un attribut `prix`.

Quel est le GRASP justifiant ce choix?

- A. Expert
- B. Couplage faible
- C. Créateur
- D. Cohésion forte



Socrative → ETSLOG210

Dans quels scénarios suivants est-ce que la responsabilité est cohérente avec le principe GRASP
Créateur?

- A. Des objets de la classe Doigt peuvent être instanciés par la classe Main.
- B. Des objets de la classe Main peuvent être instanciés par la classe Doigt.
- C. Des objets de la classe Étudiant peuvent être instanciés par la classe Magasin.
- D. Des objets de la classe LigneVente peuvent être instanciés par la classe Vente.
- E. Des objets de la classe Vente peuvent être instanciés par la classe LigneVente.



Socrative → ETSLOG210

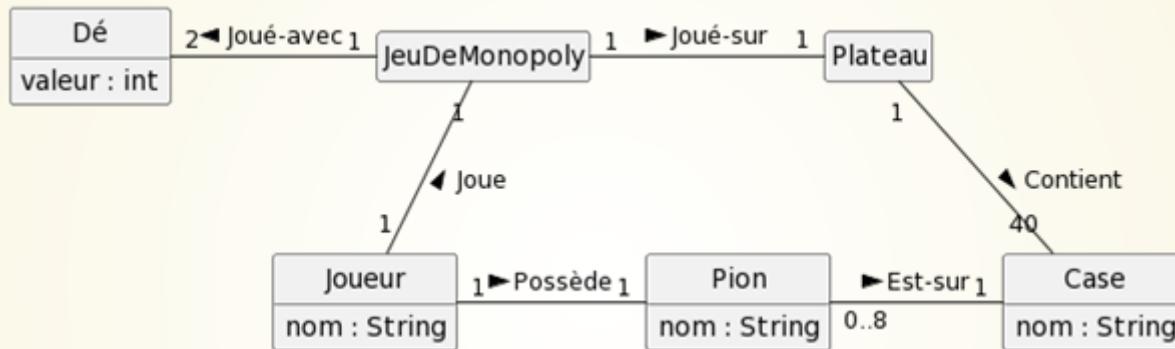
Quel principe GRASP indique qu'il faut donner une responsabilité à une classe qui possède des informations nécessaires pour s'en acquitter?

- A. Créateur
- B. Contrôleur
- C. Expert en information
- D. Fabrication Pure
- E. Polymorphisme



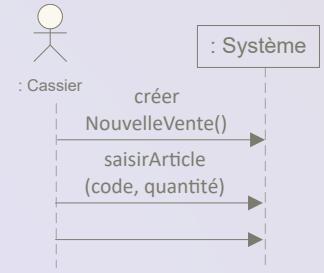
Socrative → ETSLOG210

Soit le modèle du domaine suivant:



Selon le GRASP Créateur, quelle classe devrait avoir la responsabilité de créer les objets Case?

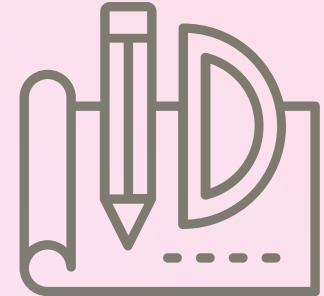
- A. La classe Plateau, puisqu'elle agrège les objets Case.
- B. La classe Pion, puisqu'elle utilise étroitement les objets Case.
- C. La classe JeuDeMonopoly, puisqu'elle possède des informations pour initialiser les objets Case.



RETOUR EXERCICE

TP#3

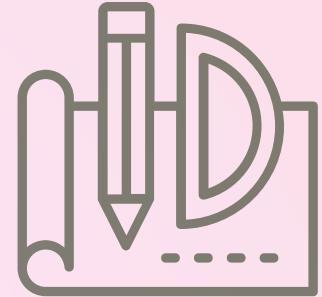
Google Classrooms ou Google Drive



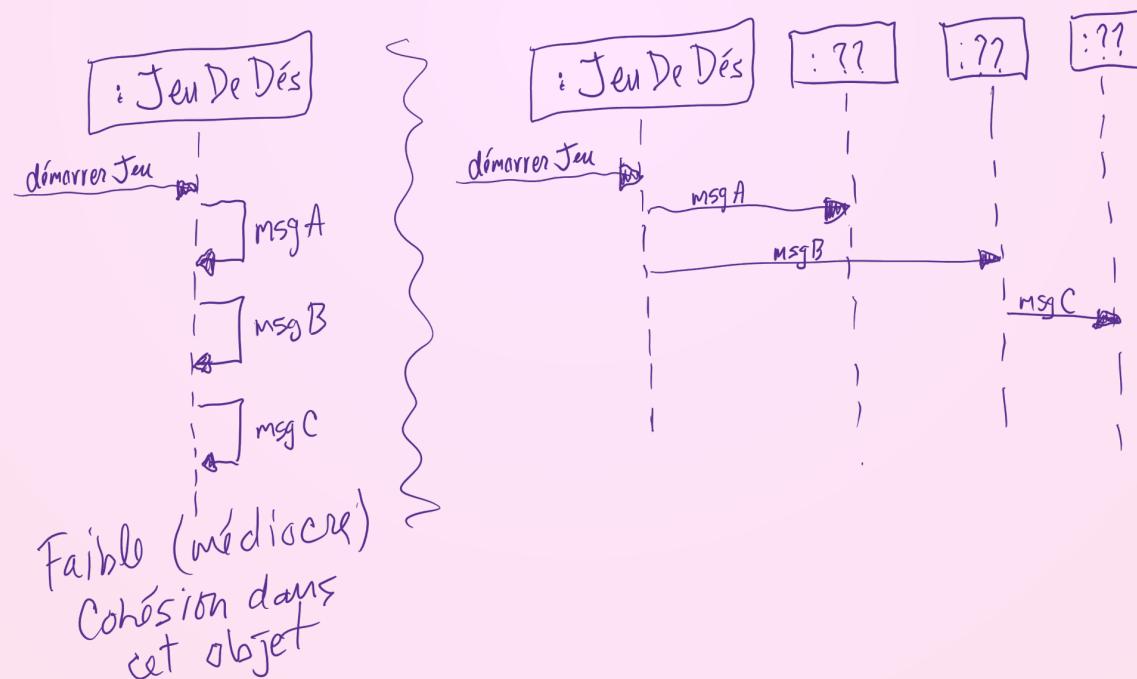
RDCU

RÉALISATION D'UN CAS D'UTILISATION

RDCU



Prendre des bonnes décisions pour une solution modulaire et facile à comprendre



DÉCALAGE DES REPRÉSENTATIONS

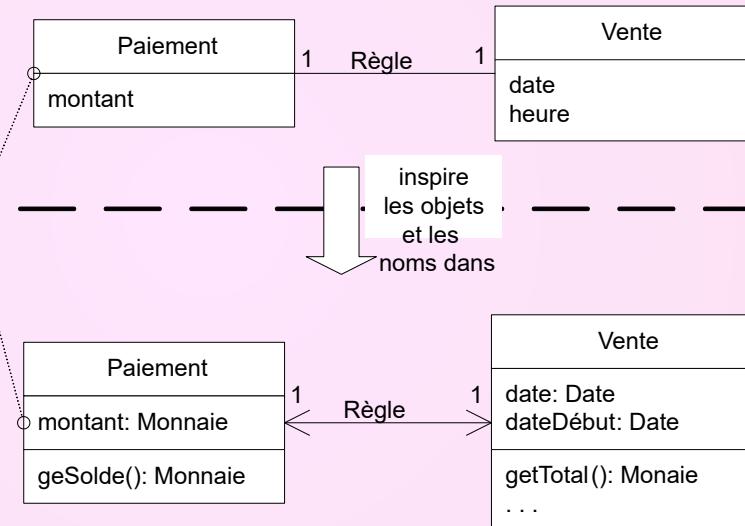


Modulaire: qui fait quoi? Qui a quelle responsabilité?

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.

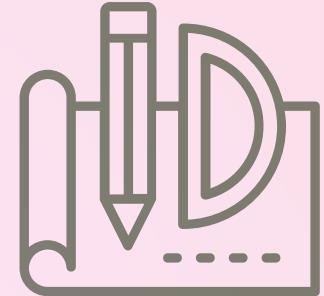


Modèle de Conception du Processus Unifié
Le développeur orienté objet s'est inspiré du domaine du monde réel pour créer les classes logicielles.

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

Facile: La solution ressemble au problème (modèles)

RDCU



Approche: conception orientée-responsabilités

GRASP

General Responsibility Assignment Software Patterns

Pour décider où mettre les méthodes...

GRASP

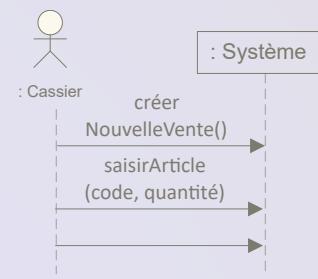
- **Contrôleur** (séparation des couches)
- **Créateur**
- **Expert en information**
- Faible couplage
- Forte cohésion
- Polymorphisme
- Indirection
- Protection des variations
- Fabrication pure



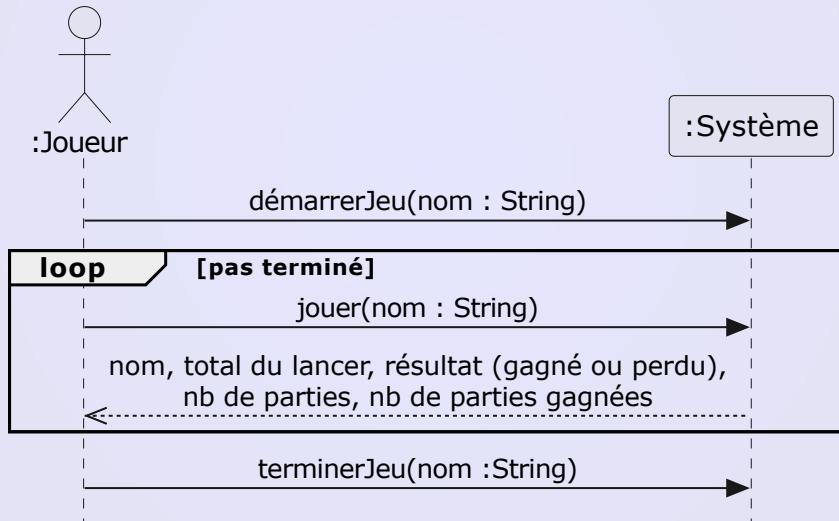
CONTRÔLEUR

RAPPEL - JEU DE DÉS

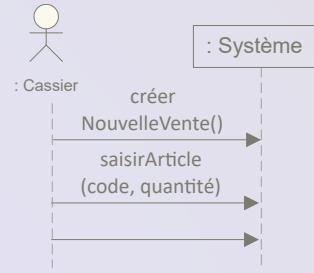
Opérations système du DSS



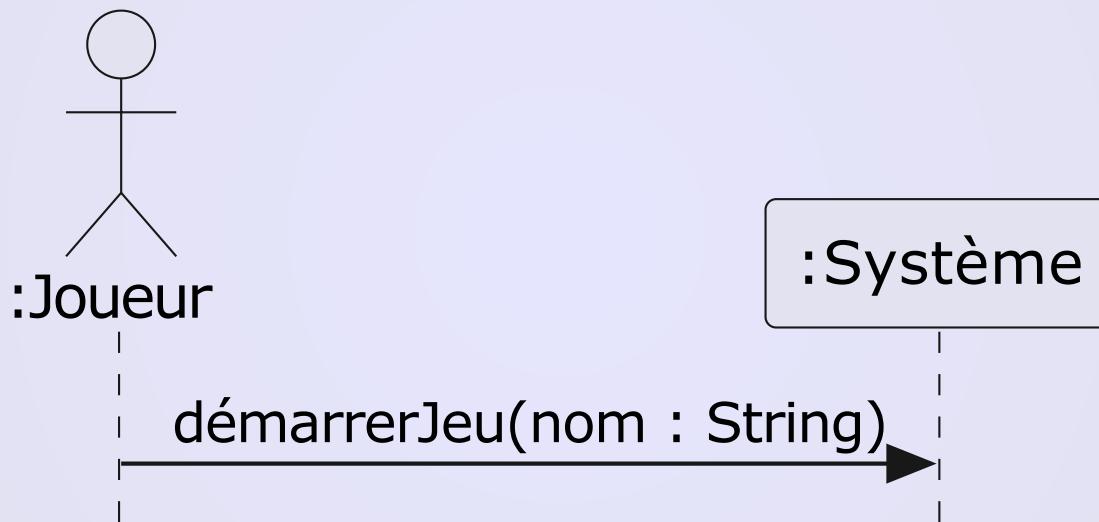
DSS pour un scénario adapté de *Jouer aux dés*
(Ch. 1 de Larman)



JEU DE DÉS



Première opération système:

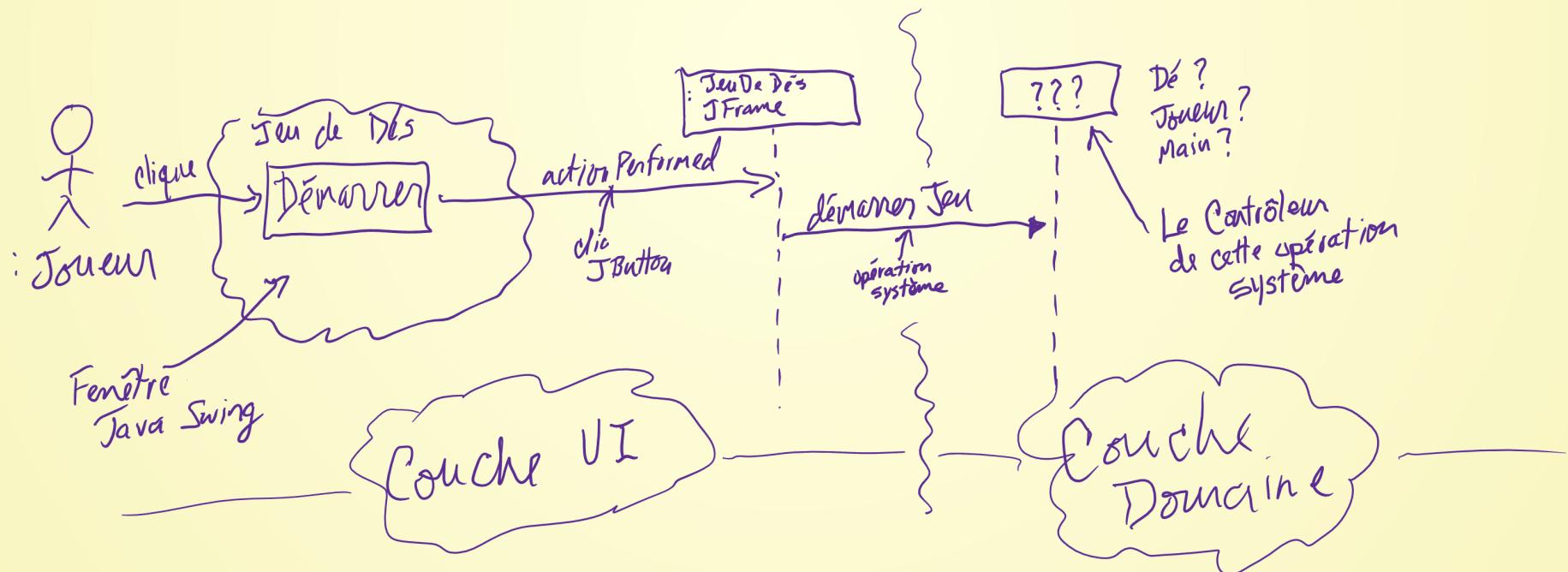


1. Qui envoie l'opération?
2. Qui la reçoit?

OPÉRATION SYSTÈME (SOLUTION JAVA SWING)



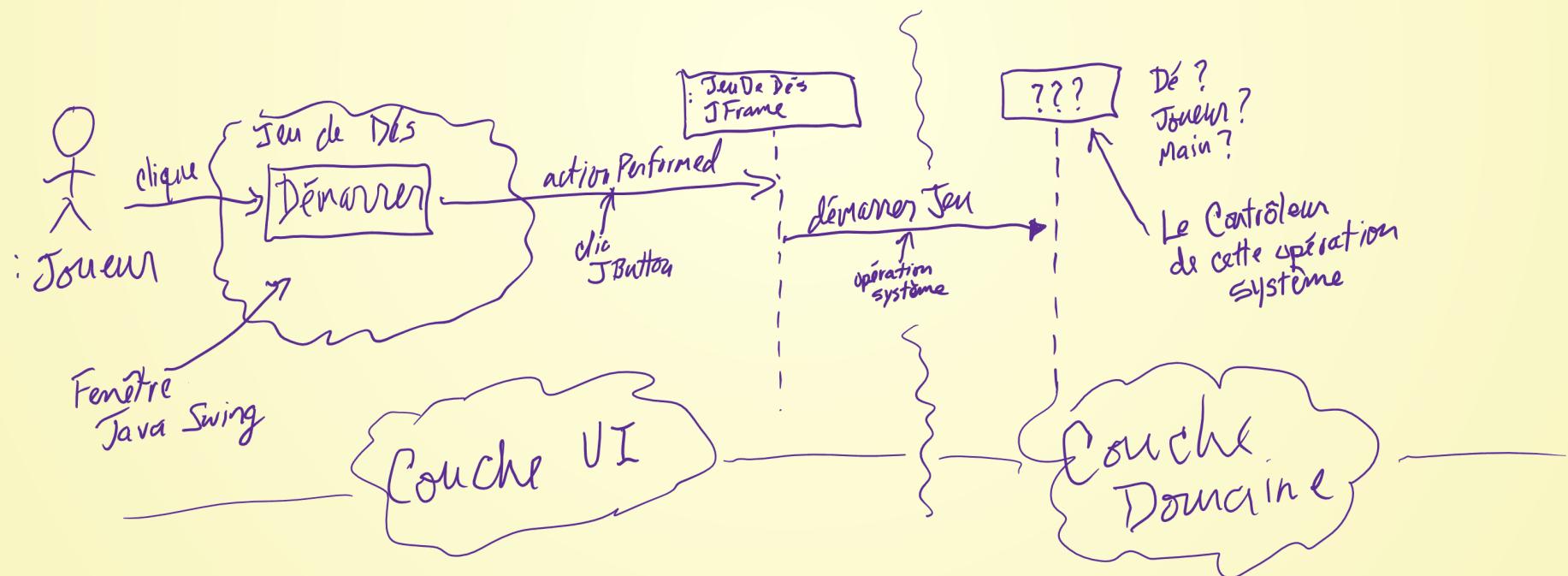
Qui envoie démarrerJeu ?



OPÉRATION SYSTÈME (SOLUTION JAVA SWING)



Qui reçoit démarrerJeu ?



PRINCIPE CONTRÔLEUR GRASP

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

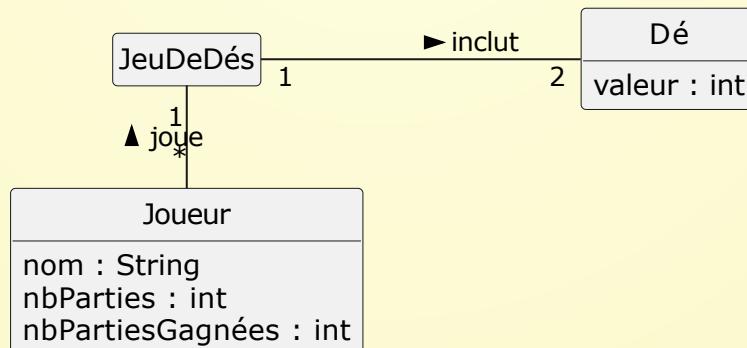
PRINCIPE CONTRÔLEUR GRASP



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)



EXEMPLES

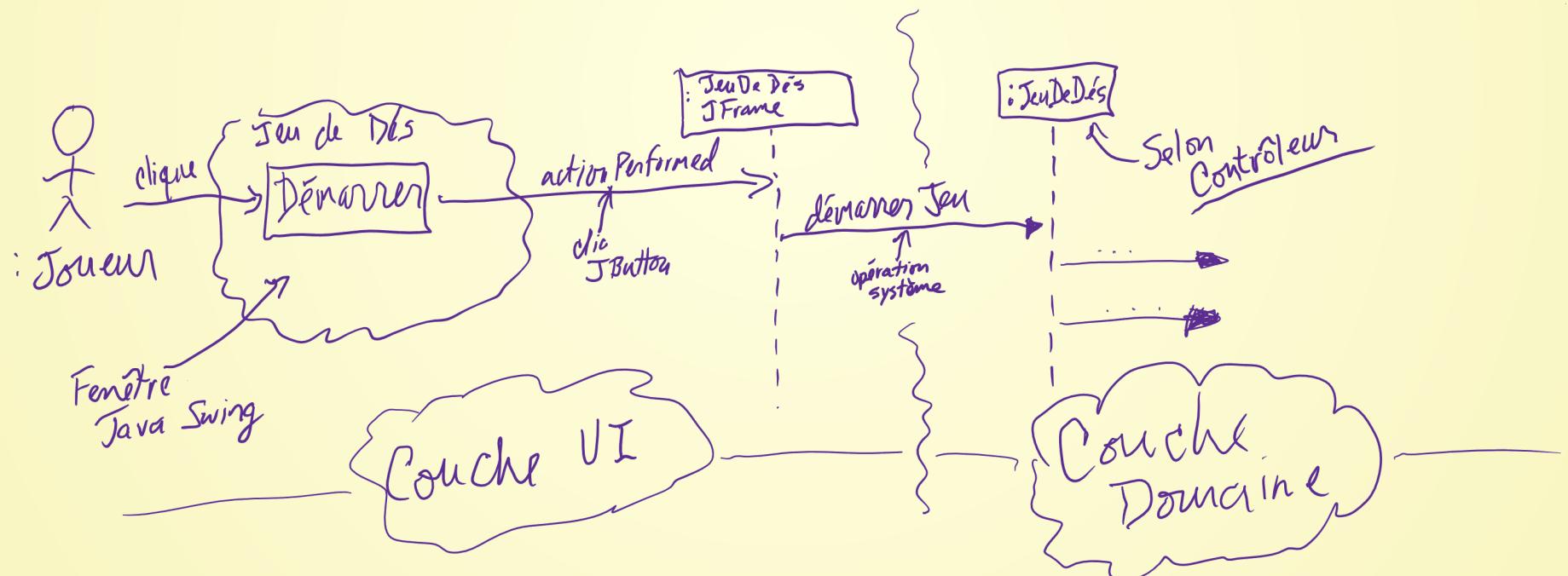
- système globale: Bixi, Cheminot, Rubik, JeuDeDés
- objet racine: l'objet qui relit la plupart des autres (Université, Bibliothèque, JeuDeDés)
- équipement: utilisé dans le contexte du cas d'utilisation (Caisse, BorneBixi, etc.)

On n'est pas “surpris” d'imaginer que la classe traite l'opération système.

RDCU - CONTRÔLEUR



JeuDeDés est le contrôleur GRASP (inspiré du MDD)



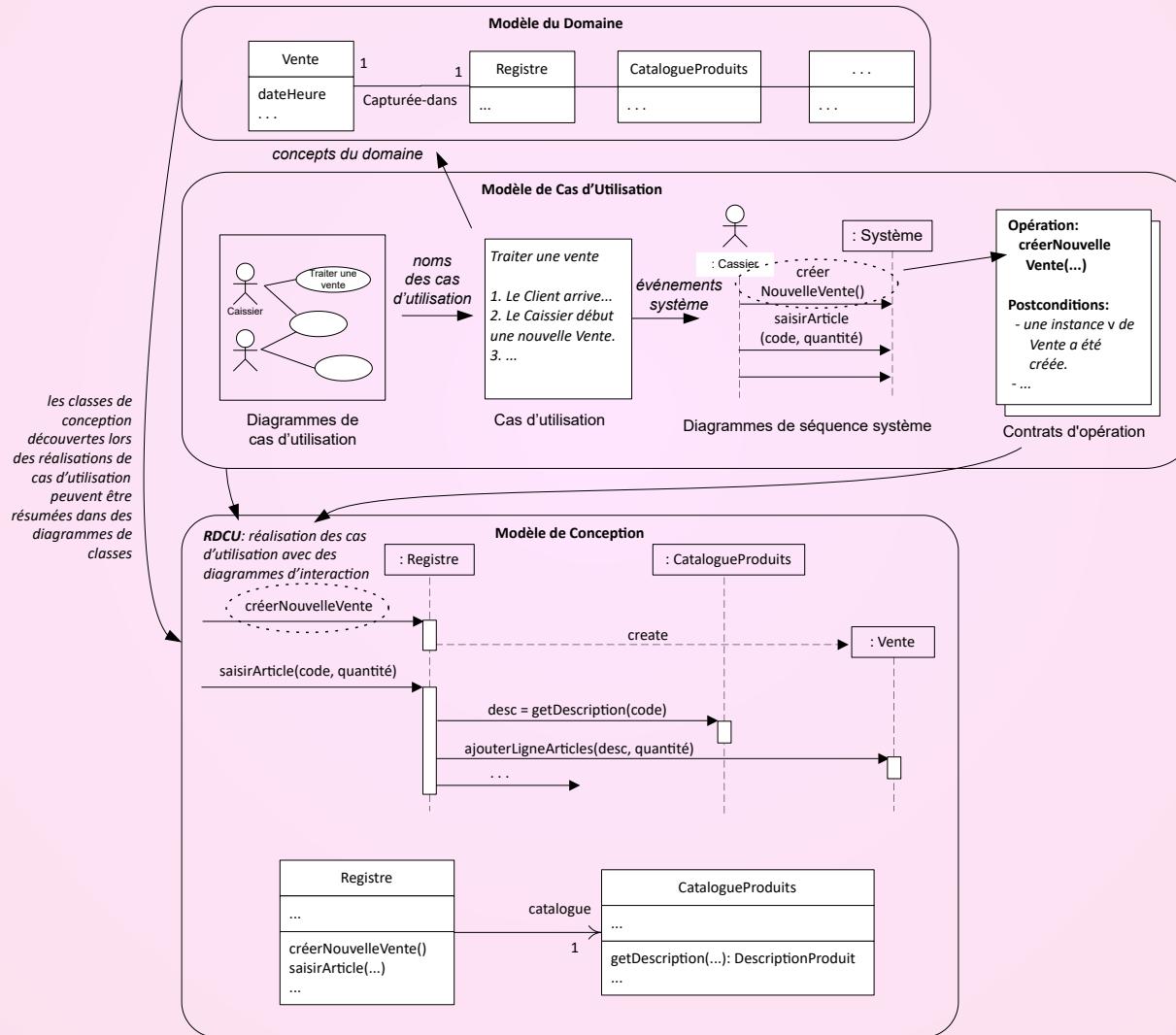
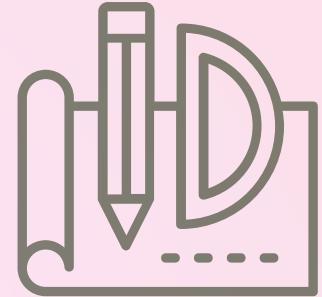
POURQUOI LES OPÉRATION SYSTÈME ET LES CONTRÔLEURS?

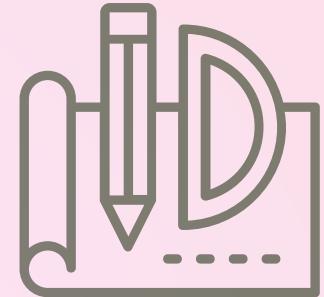
Cela favorise la bonne séparation des couches.



CRÉATEUR

RDCU (SURVOL)

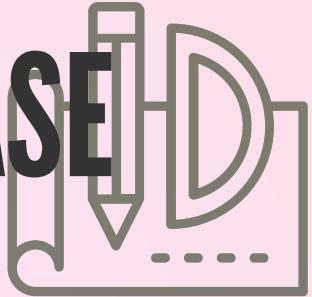




RDCU: SCÉNARIO DÉMARRER

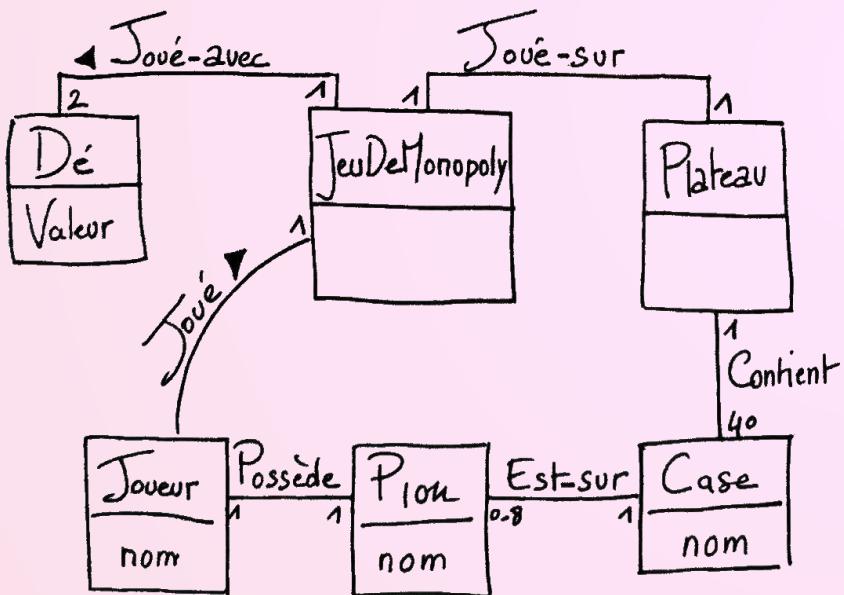
- C'est l'initialisation du système.
- C'est implicite mais essentiel!
- On doit instancier les objets faisant partie de l'application.
- Exemple: Monopoly - instancier les objets Case

QUI INSTANCIE LES OBJETS CASE

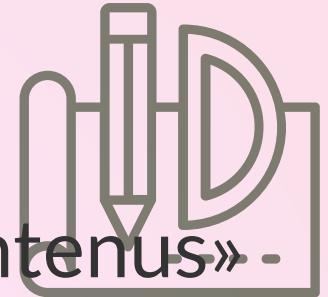


Socrative → ETSLOG210

- A. Dé
- B. JeuDeMonopoly
- C. Plateau
- D. Joueur
- E. Pion



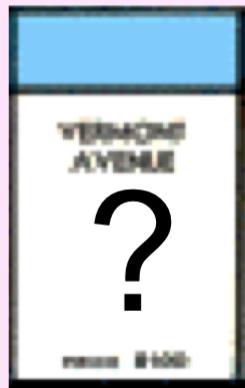
CRÉATEUR (GRASP)



- Les «conteneurs» créent les objets «contenus» -



Chien



Case

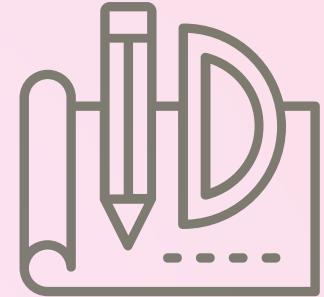


Plateau



Dé

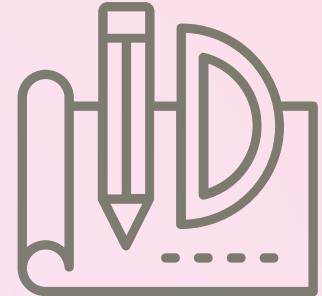
CRÉATEUR (GRASP)



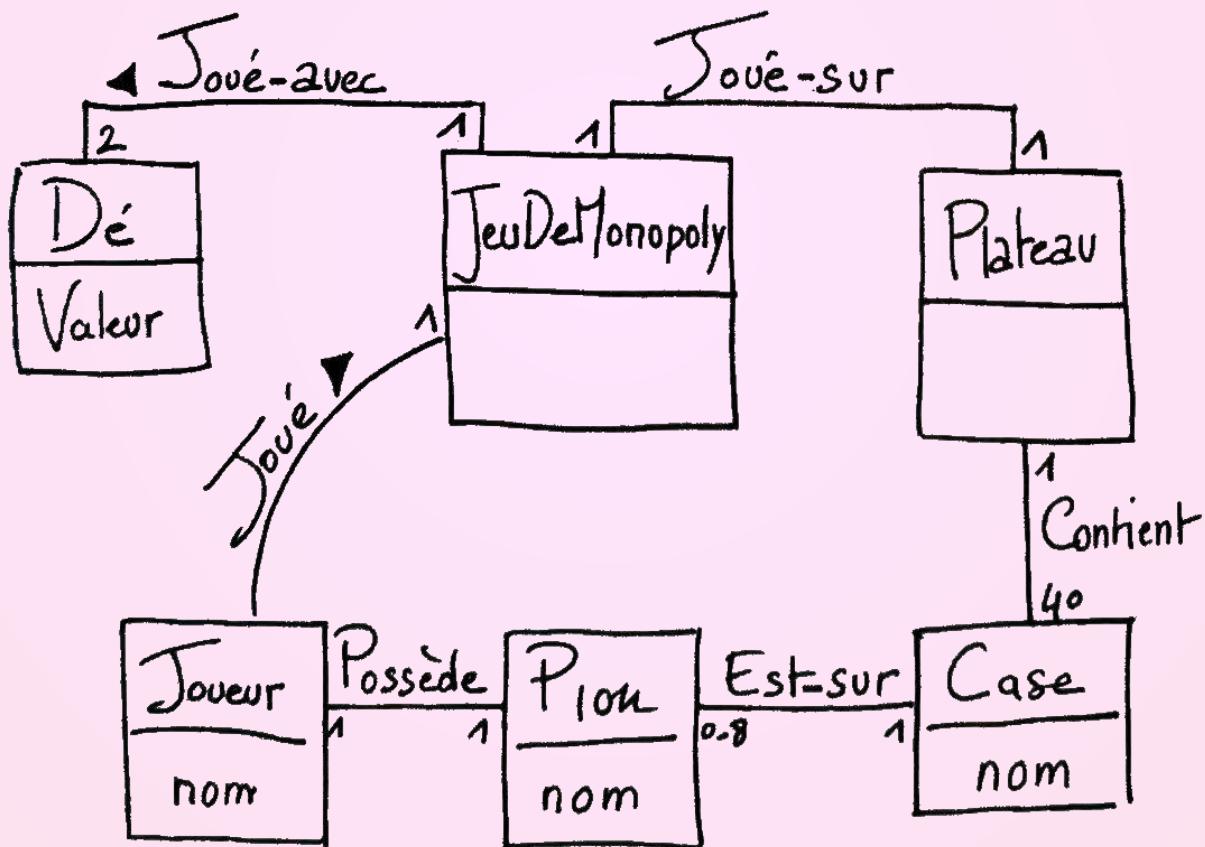
- **Problème:** Qui crée? (postcondition d'un contrat)
- **Solution:** Affecter à la classe B la responsabilité de créer les objets d'une classe A si...
 - B possède les données d'initialisation des objets A
 - B contient ou agrège des objets A
 - B utilise étroitement des objets A
 - B enregistre des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

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

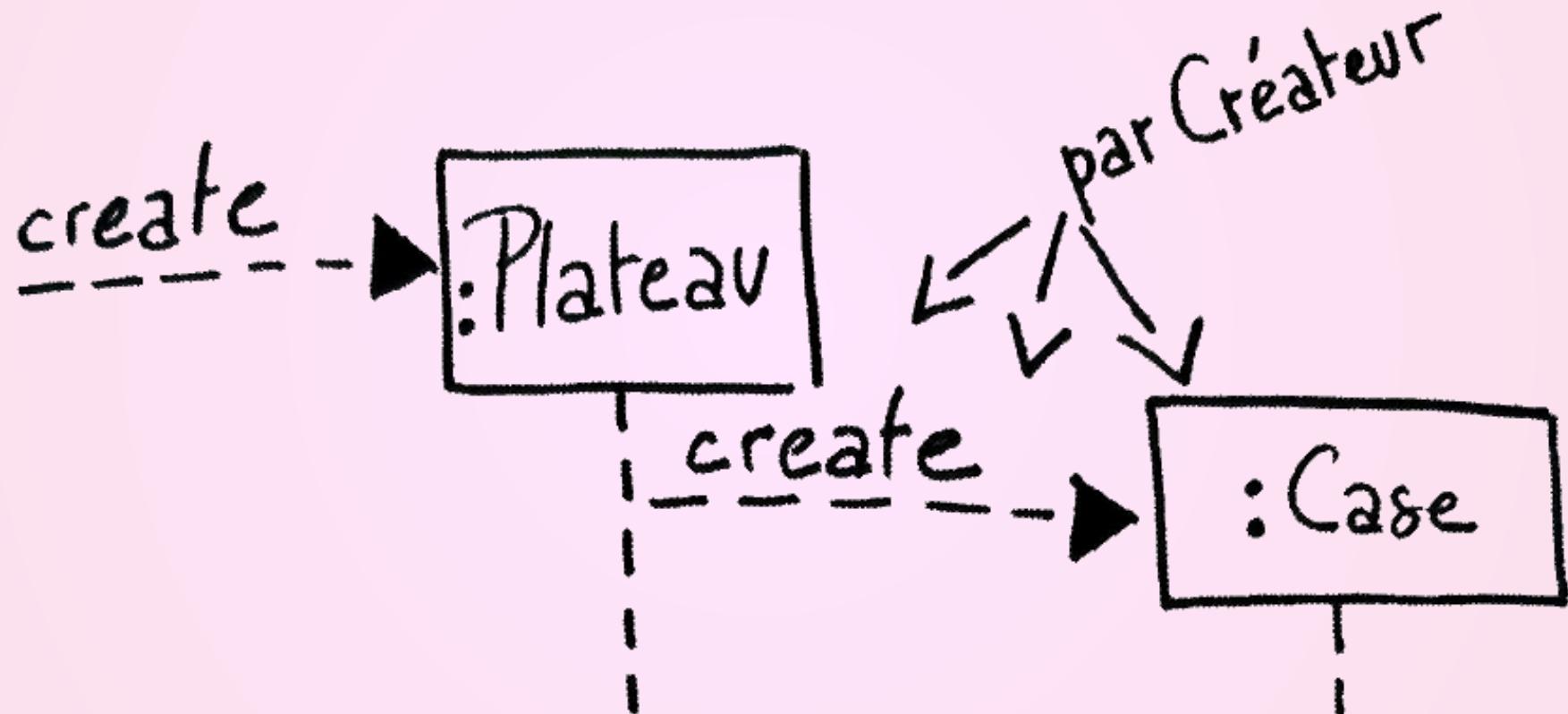
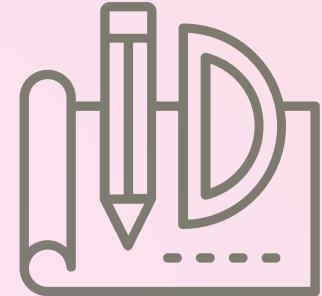
CRÉATEUR



- Qui crée les cases (Square)?



CRÉATEUR (ANNOTATION)



EXPERT EN INFORMATION

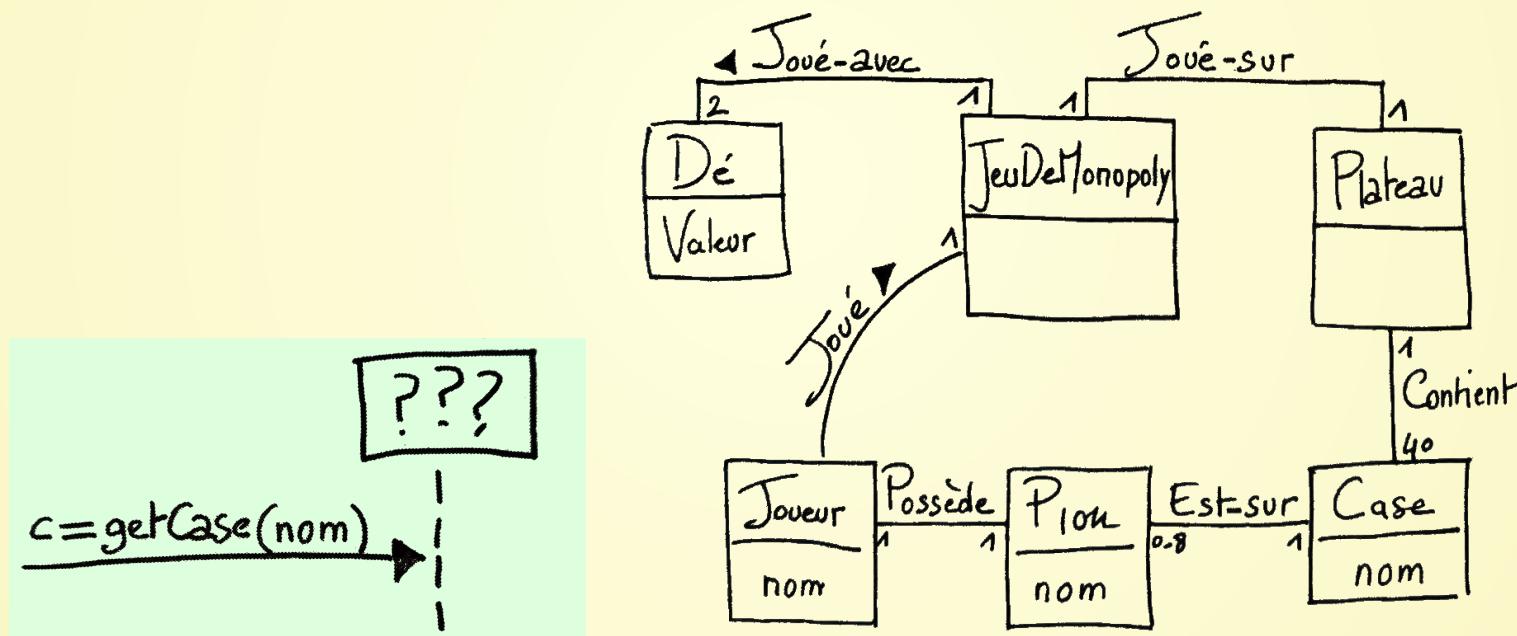
EXPERT EN INFORMATION

- **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

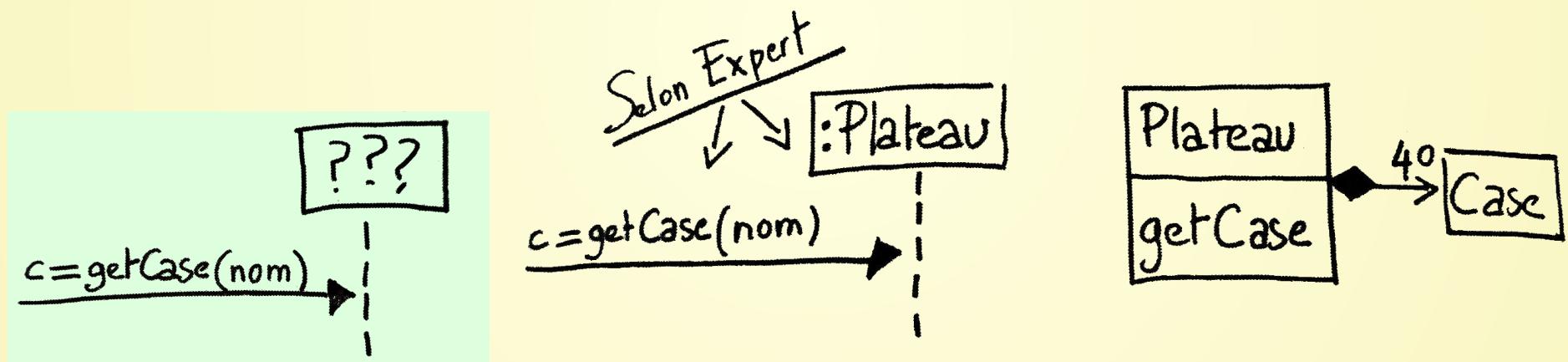
EXPERT (GRASP)

Où mettre la méthode `getCase(nom)`?

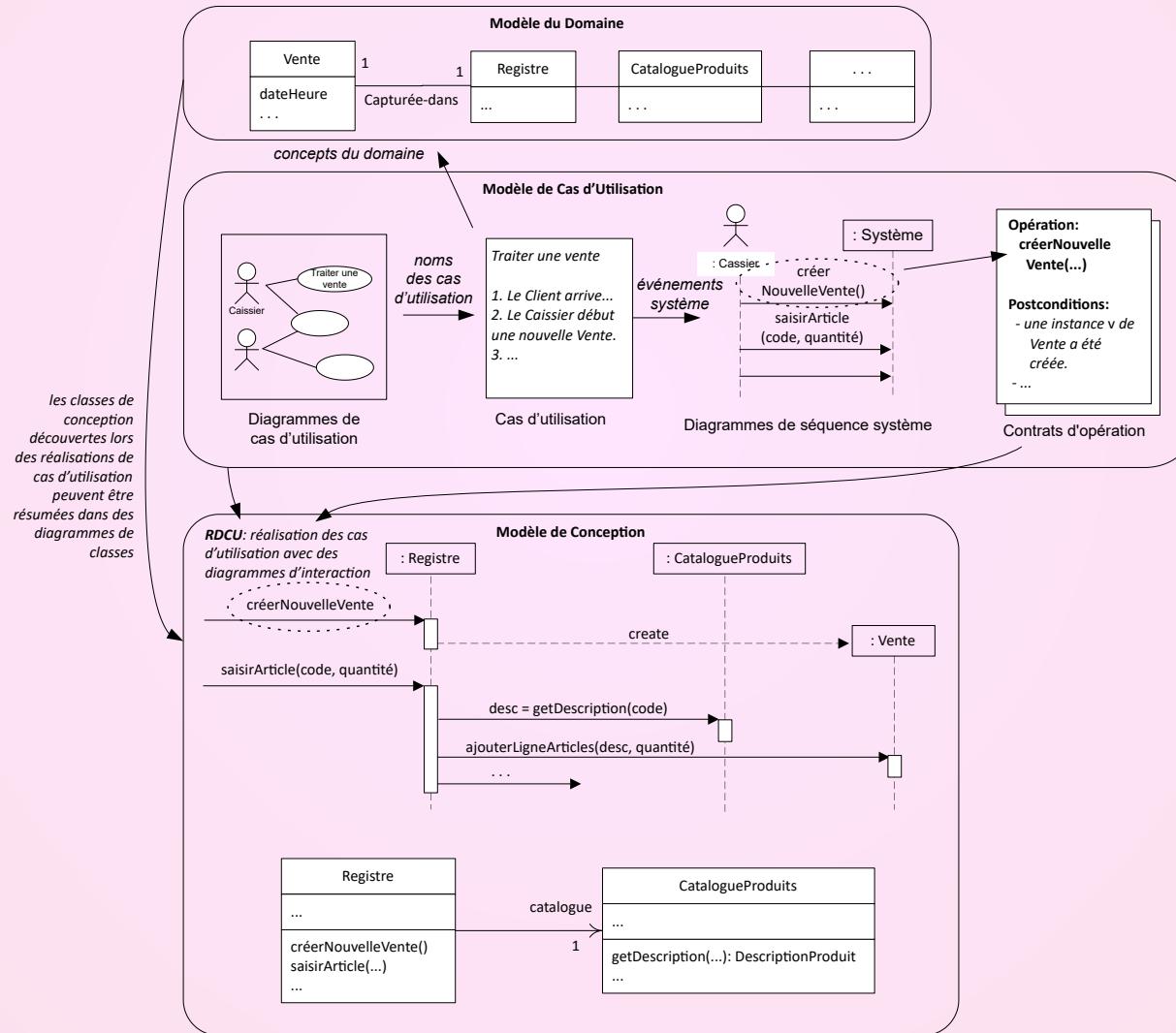
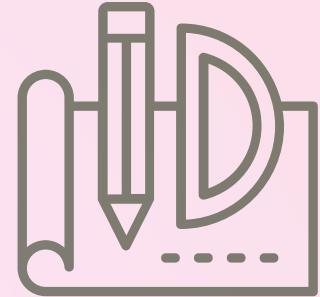


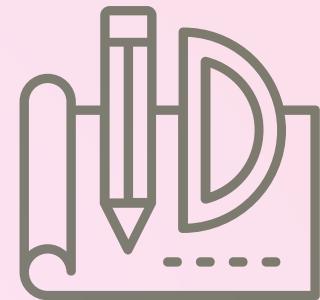
EXPERT

Application du patron Expert



RDCU (SURVOL)

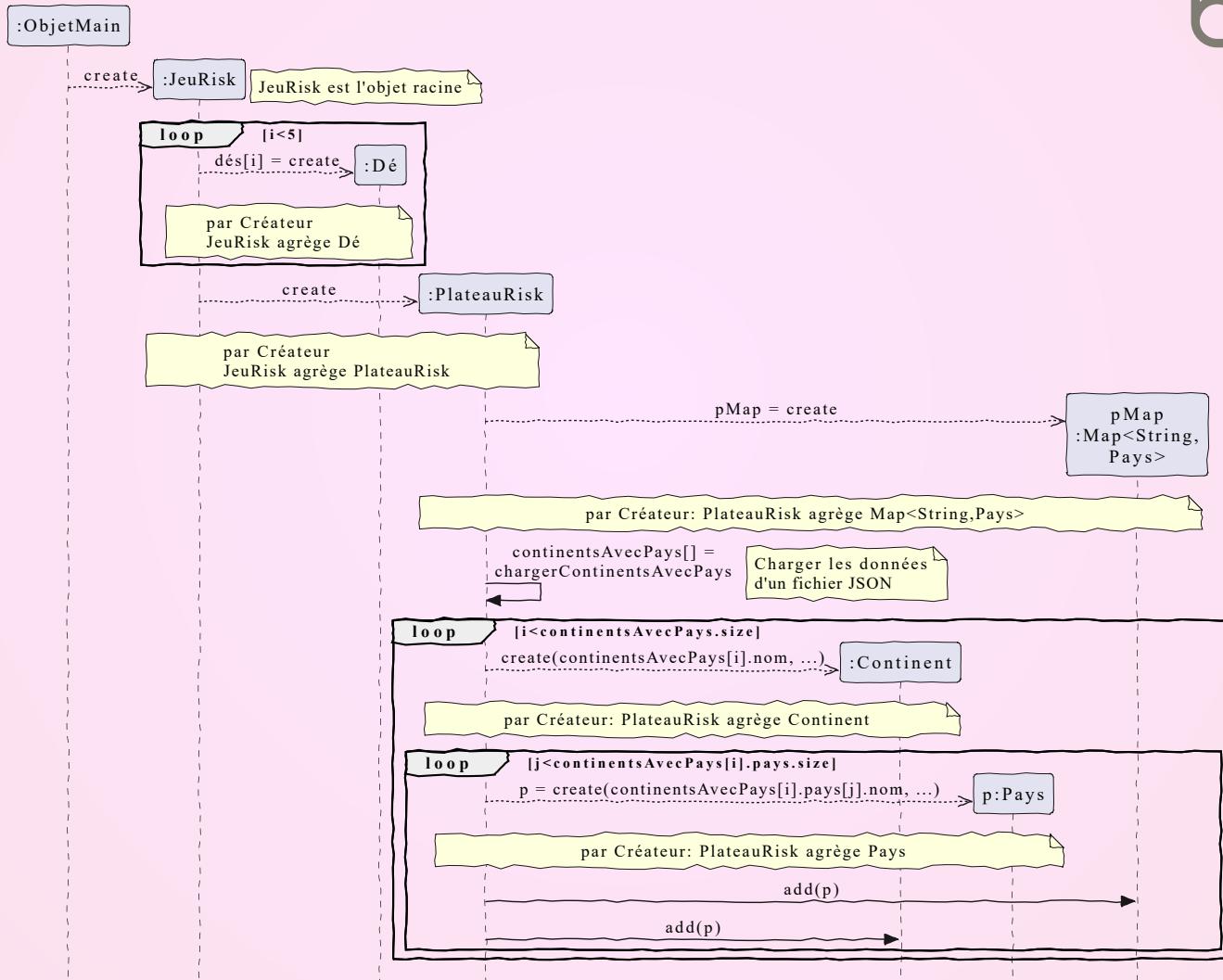
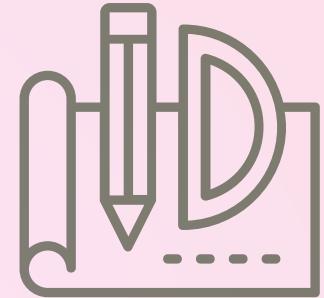




RDCU: SCÉNARIO DÉMARRER

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

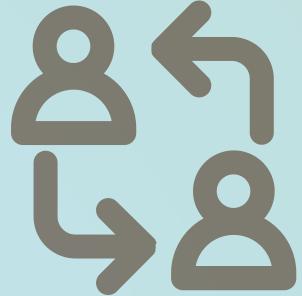
SCÉNARIO DÉMARRER (RISK - MDD)



RÉSUMÉ GRASP

Vulgarisation!

- Contrôleur - à qui envoyer l'opération système
- Créeateur - qui fait new X()
- Expert - qui est le plus capable de faire y()



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.