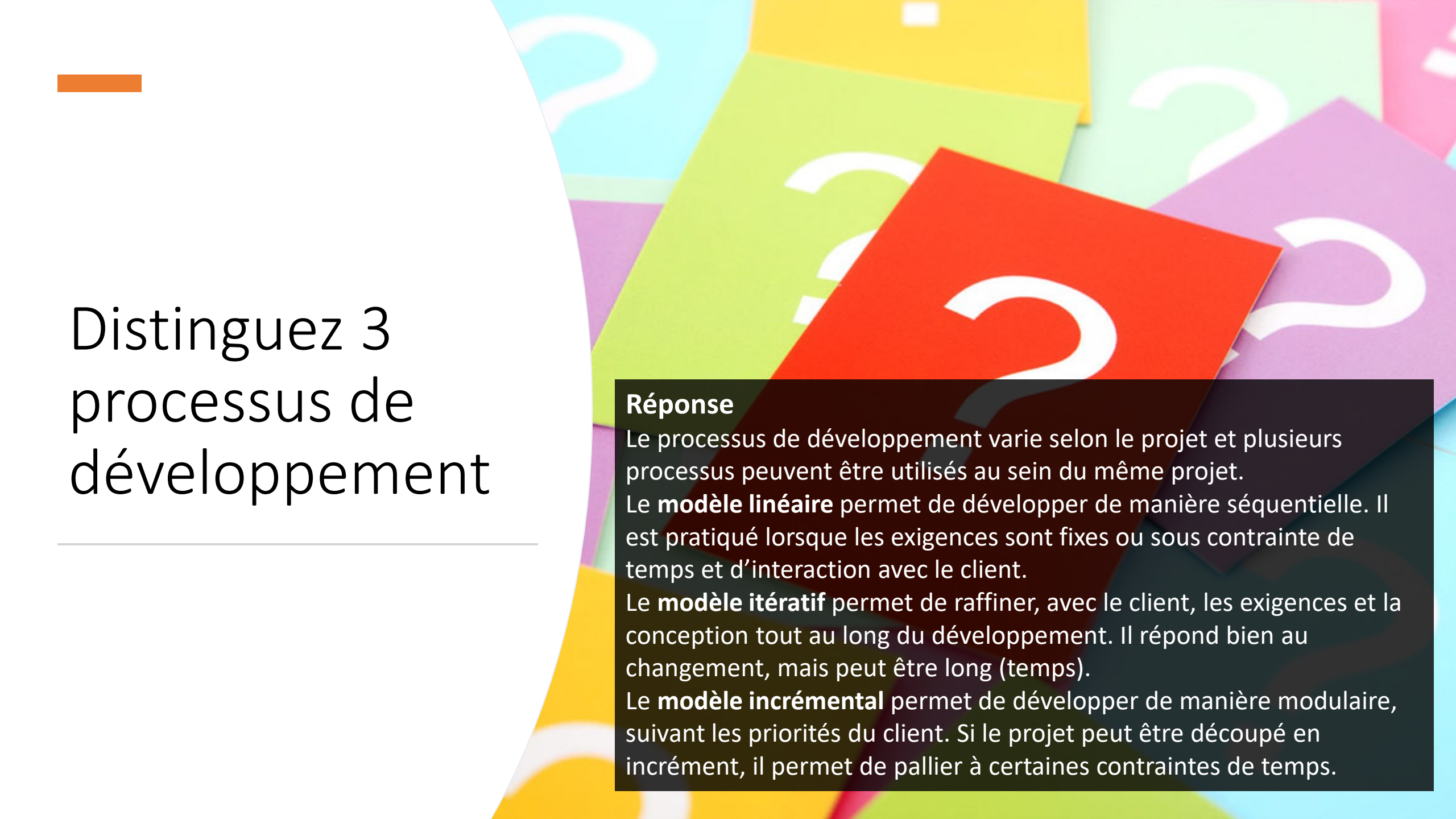


# Révision - Final

---

IFT2255 – Génie logiciel

Louis-Edouard LAFONTANT



# Distinguez 3 processus de développement

---

## Réponse

Le processus de développement varie selon le projet et plusieurs processus peuvent être utilisés au sein du même projet.

Le **modèle linéaire** permet de développer de manière séquentielle. Il est pratiqué lorsque les exigences sont fixes ou sous contrainte de temps et d'interaction avec le client.

Le **modèle itératif** permet de raffiner, avec le client, les exigences et la conception tout au long du développement. Il répond bien au changement, mais peut être long (temps).

Le **modèle incrémental** permet de développer de manière modulaire, suivant les priorités du client. Si le projet peut être découpé en incrément, il permet de pallier à certaines contraintes de temps.

Un besoin peut se traduire en fonctionnalité ou en contrainte.

---



### Réponse

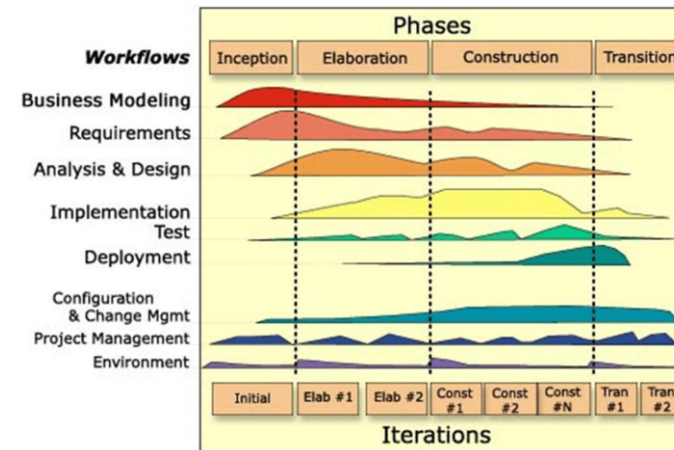
Les besoins formulés par le client servent à déterminer les tâches et fonctions que doit supporter la solution ainsi que les paramètres rendant la solution acceptable dans son contexte d'exploitation. Un besoin peut ainsi donner lieu à une fonctionnalité (tâche à accomplir, décrite par les CUs) ou une contrainte sur les caractéristiques du logiciel ou l'une de ces fonctionnalités.

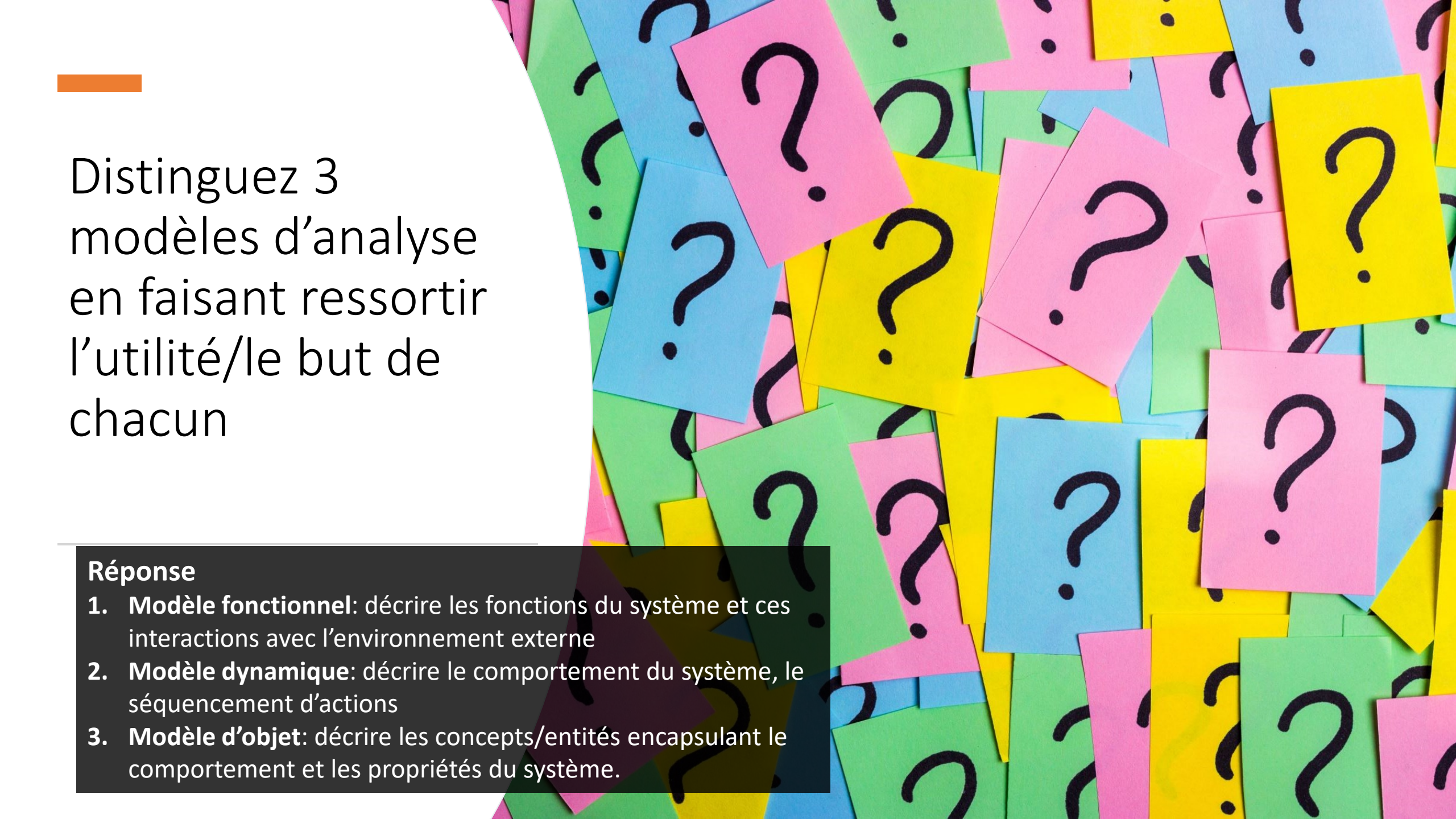
Le modèle unifié suit un processus itératif et incrémental



### Réponse

Le processus unifié découpe un projet en plusieurs phases au travers desquels on itère sur des flux de développement (à différent degré).

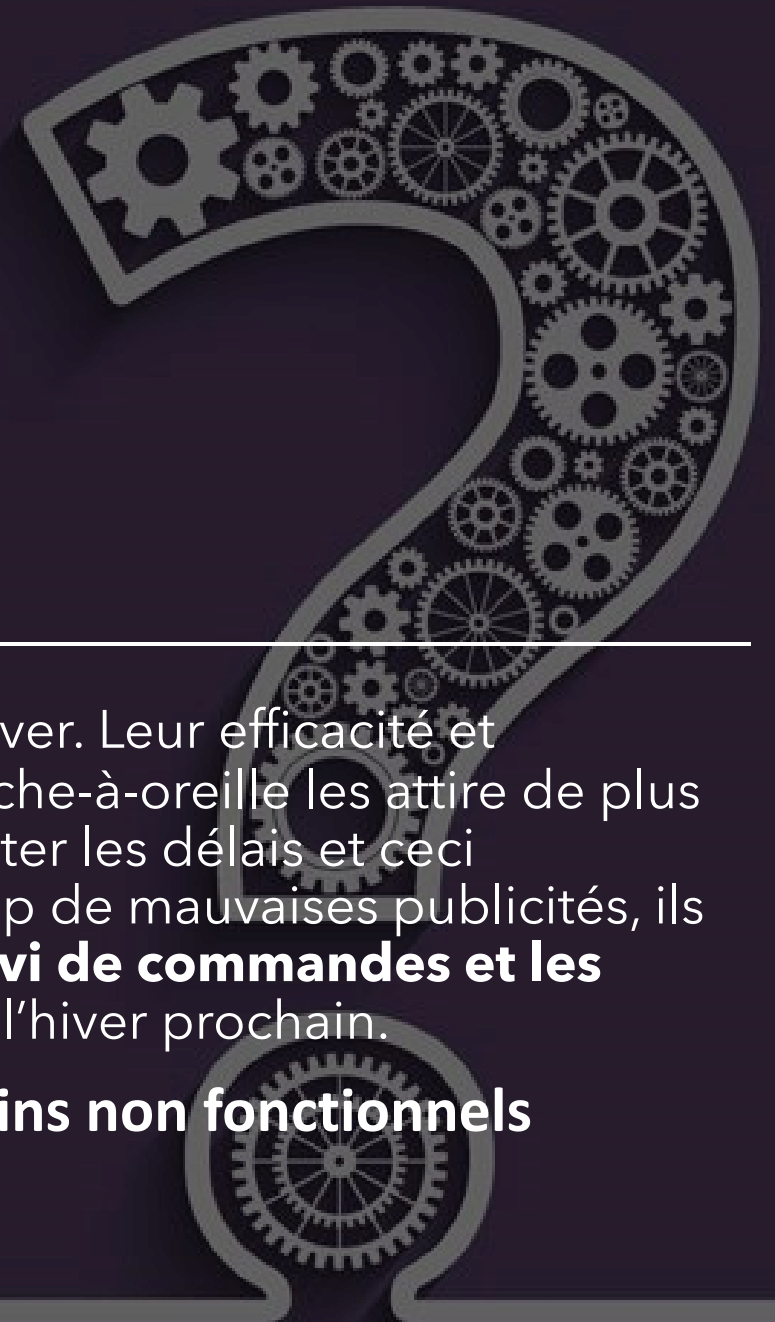


The background of the slide is a collage of colorful sticky notes in shades of pink, yellow, blue, and green. Each sticky note has a large black question mark printed on it. The notes are scattered and overlapping, creating a sense of inquiry and confusion. On the left side, there is a white curved shape that frames the text.

Distinguez 3  
modèles d'analyse  
en faisant ressortir  
l'utilité/le but de  
chacun

### Réponse

1. **Modèle fonctionnel:** décrire les fonctions du système et ces interactions avec l'environnement externe
2. **Modèle dynamique:** décrire le comportement du système, le séquençage d'actions
3. **Modèle d'objet:** décrire les concepts/entités encapsulant le comportement et les propriétés du système.



---

L'équipe **Paraneige** installe des abris autos temporaires à l'hiver. Leur efficacité et convivialité a impressionné un nombre de clients, et le bouche-à-oreille les attire de plus en plus de commandes. Cependant, ils ont du mal à respecter les délais et ceci commence à ternir l'image de la compagnie. Pour éviter trop de mauvaises publicités, ils font appel à vous pour trouver un **meilleur système de suivi de commandes et les permettant de coordonner leurs tâches**, d'ici le début de l'hiver prochain.

**Identifiez 3 risques et 3 besoins fonctionnels et 3 besoins non fonctionnels**



### Réponse

Une bonne conception se traduit par un **faible couplage et une forte cohésion**, où le couplage représente le degré d'interaction entre les modules et la cohésion le degré d'interaction au sein du module. Ceci favorise la **modularité** et **flexibilité** du système et permet de comprendre et modification le système avec plus d'aisance. Elle doit aussi se **conformer aux spécifications** du projet établi lors de l'analyse des exigences.

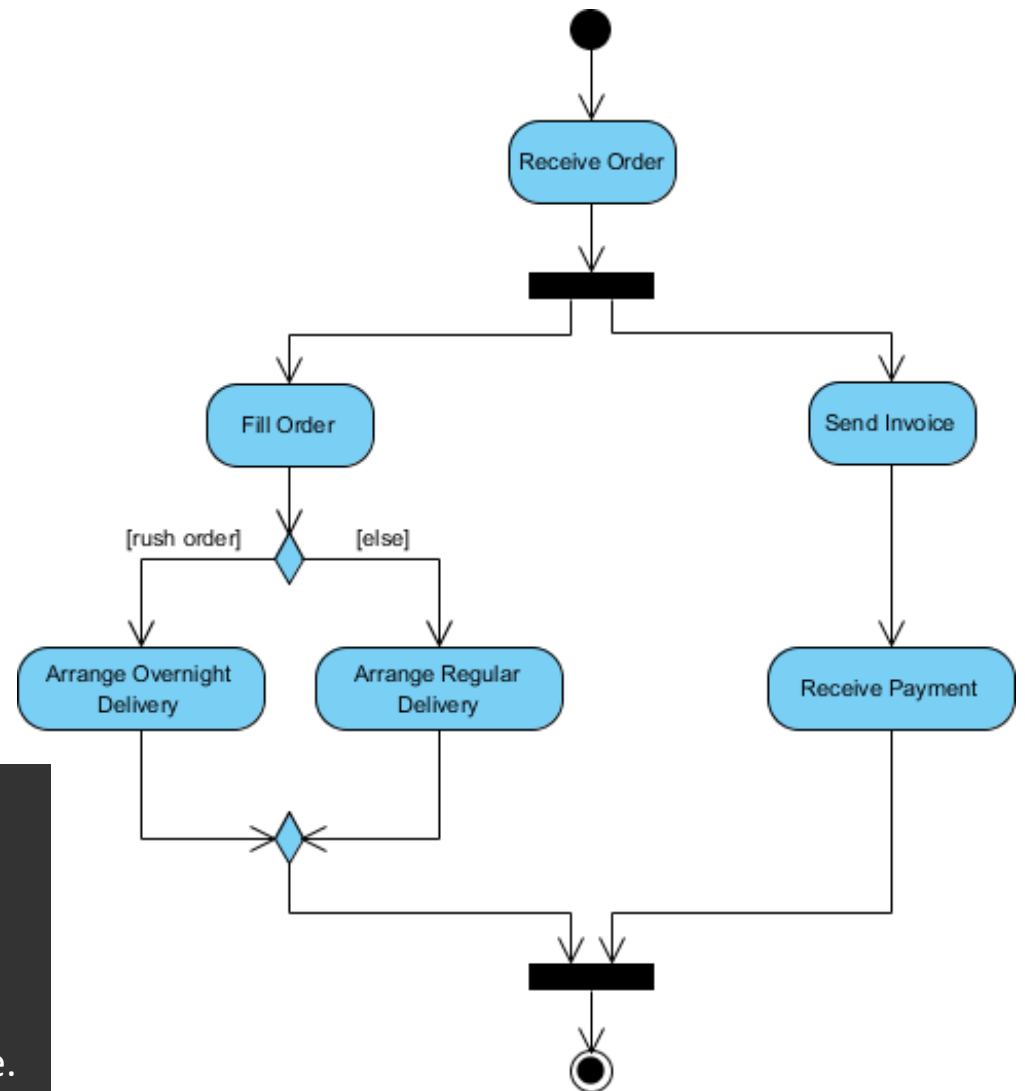
Quelles sont les caractéristiques d'une bonne conception?



Décrivez le flux dans le diagramme d'activité suivant

### Réponse

Suite à la réception d'une commande, le traitement de la facturation et de la commande s'effectue en parallèle. Le traitement de la facturation consiste à envoyer la facture et recevoir le paiement. Le traitement de la commande consiste à remplir le bon de commande et choisir l'option de livraison (*rush*). Dépendamment de l'option choisie, on prépare une livraison rapide ou régulière. Lorsque les deux traitements sont terminés, la commande est complétée.







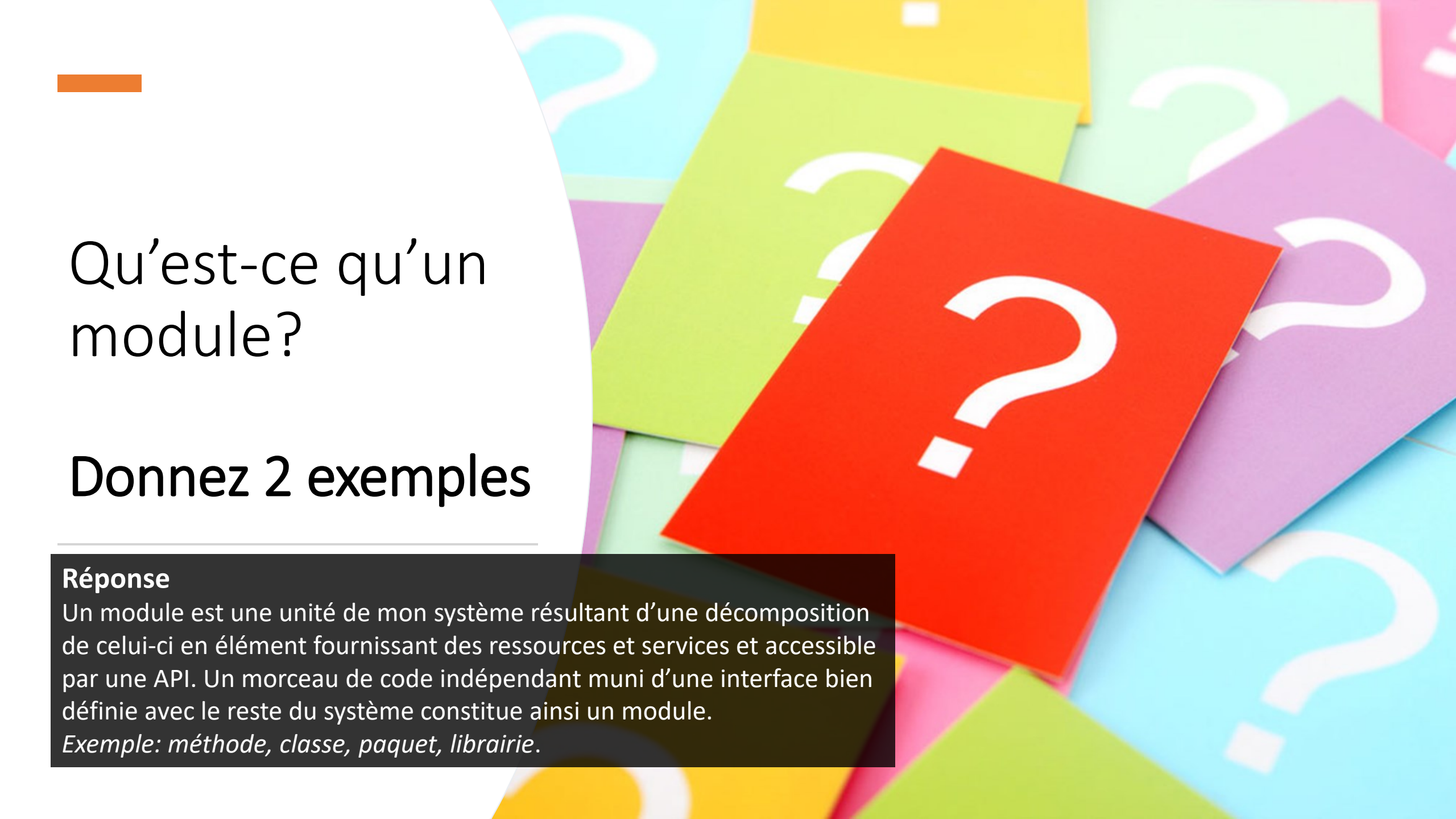
UML est un langage  
de programmation

---



### Réponse

UML est un langage de modélisation composé de 14 types de diagrammes (structuraux et comportementaux). Il peut être couplé à des techniques de génération de code pour créer des programmes (exécutables).



—

Qu'est-ce qu'un module?

Donnez 2 exemples

**Réponse**

Un module est une unité de mon système résultant d'une décomposition de celui-ci en élément fournissant des ressources et services et accessible par une API. Un morceau de code indépendant muni d'une interface bien définie avec le reste du système constitue ainsi un module.

*Exemple: méthode, classe, paquet, librairie.*

—

L'agrégation  $A \langle \rangle \text{--} B$  implique que si A est détruit, B sera aussi détruit

---



### Réponse

Une relation d'agrégation entre deux classes signale que les objets d'une classe sont des composants de l'autre classe. Elle permet de définir des objets composés d'autres objets. Cependant, dans la relation de **composition**, chaque composant est une partie d'un seul composite. Si le composite est détruit (ou copié), ses composants le sont aussi.

---

# Qu'est-ce que le polymorphisme?

## Réponse

Le polymorphisme consiste à faire varier le comportement d'une opération en la définissant dans plus d'une classe héritant d'une même classe (parent, abstrait en général).



Avec un framework,  
je suis responsable  
de la logique de  
contrôle

---



#### Réponse

Dans un *framework*, la **plateforme incorpore la logique de contrôle générique**. Le code développé doit donc se conformer à la logique de contrôle du *framework*.  
En revanche, dans une librairie, l'utilisateur de la librairie est responsable de la logique de contrôle.

Quelle est la différence entre une architecture MVC et une architecture 3-tier?

**Réponse**

L'architecture MVC divise une application en 3 parties interconnectées: Modèle, Vue, Contrôleur. Le modèle communique son état à la vue chargé de la présentation. Le contrôleur traite les interactions reçues par la vue et envoie des requêtes au modèle.

L'architecture 3-tier divise un système en 3 couches distinctes, où une couche ne peut communiquer qu'avec ces couches voisines.


Le développement d'interfaces est crucial pour la programmation distribuée

---



### Réponse

Les interfaces forment des contrats entre les modules. Elles permettent de paralléliser le développement. Chaque équipe peut progresser en se fiant à l'interface qui sera implémentée par une autre équipe.



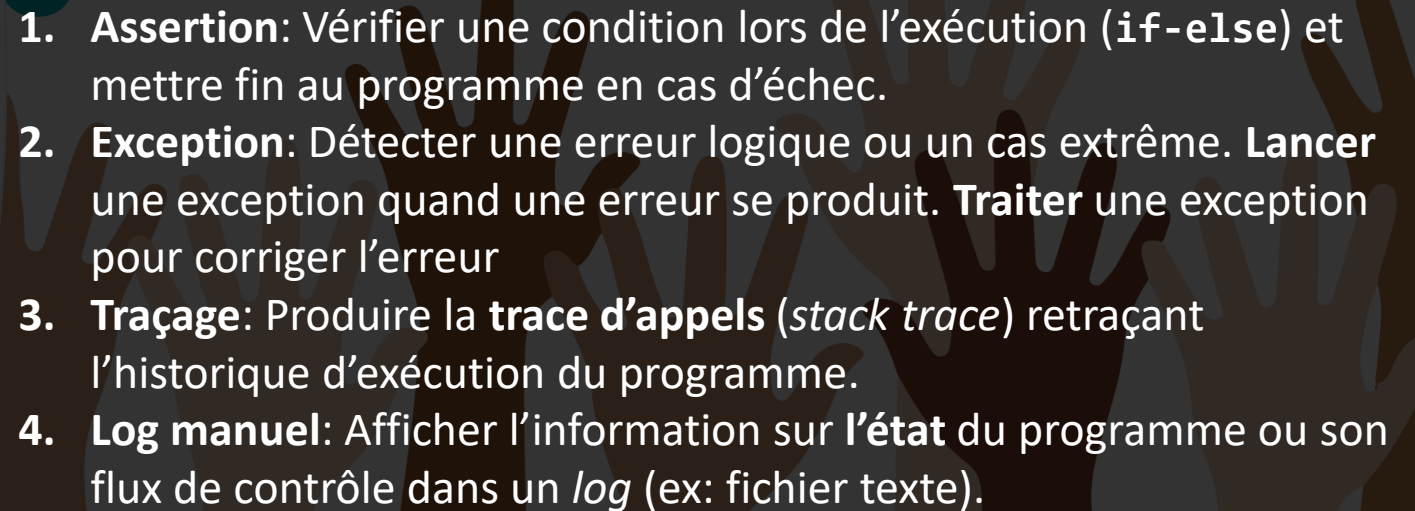
# Distinguez 3 techniques de débogage

---



## Réponse

Afin de trouver et réduire le nombre de défauts dans un programme, on peut utiliser les techniques suivantes:

1. **Assertion:** Vérifier une condition lors de l'exécution (`if-else`) et mettre fin au programme en cas d'échec.
  2. **Exception:** Détecter une erreur logique ou un cas extrême. **Lancer** une exception quand une erreur se produit. **Traiter** une exception pour corriger l'erreur
  3. **Traçage:** Produire la **trace d'appels** (*stack trace*) retraçant l'historique d'exécution du programme.
  4. **Log manuel:** Afficher l'information sur l'état du programme ou son flux de contrôle dans un *log* (ex: fichier texte).
- 




Le *refactoring* consiste à modifier les fonctionnalités implémentées dans le code.

---



### Réponse

Les opérations de refactoring **modifient la structure du code sans affecter la fonctionnalité**. Elles peuvent servir à améliorer la lisibilité du code et à le structurer afin de minimiser l'impact des changements futurs et faciliter la maintenance.

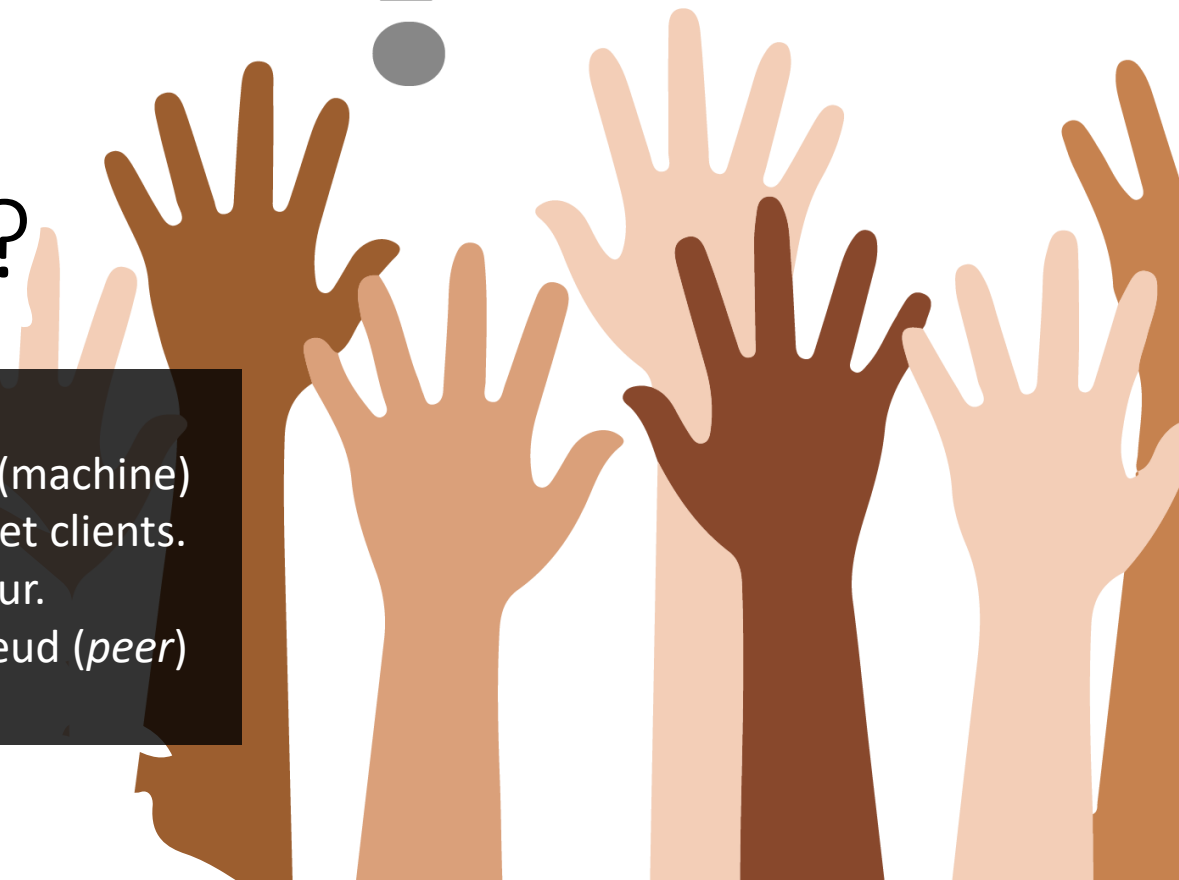


Quelle est la différence  
entre une architecture  
client-serveur et une  
architecture peer-to-peer?

### Réponse

Dans une architecture **client-serveur (centralisé\*)**, les nœuds (machine) participants à un système se divisent en deux parties: serveur et clients. Le serveur fournit des services aux clients qui contactent le serveur.

Dans une architecture **peer-to-peer (décentralisé)**, chaque nœud (*peer*) joue à la fois le rôle du client et du serveur.

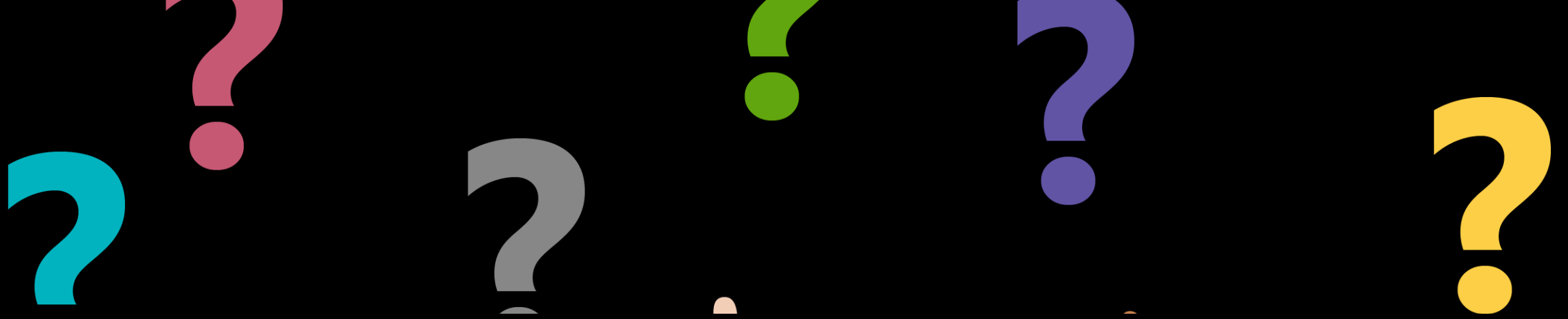


Les tests ne permettent pas de garantir l'absence d'erreur




### Réponse

Les tests permettent de **vérifier la présence d'une erreur donnée** (décrite dans le test). Aucun nombre de tests ne peut garantir le comportement entier du logiciel ou l'absence d'erreur.



Décrivez le flux de  
changements apportés post-  
déploiement





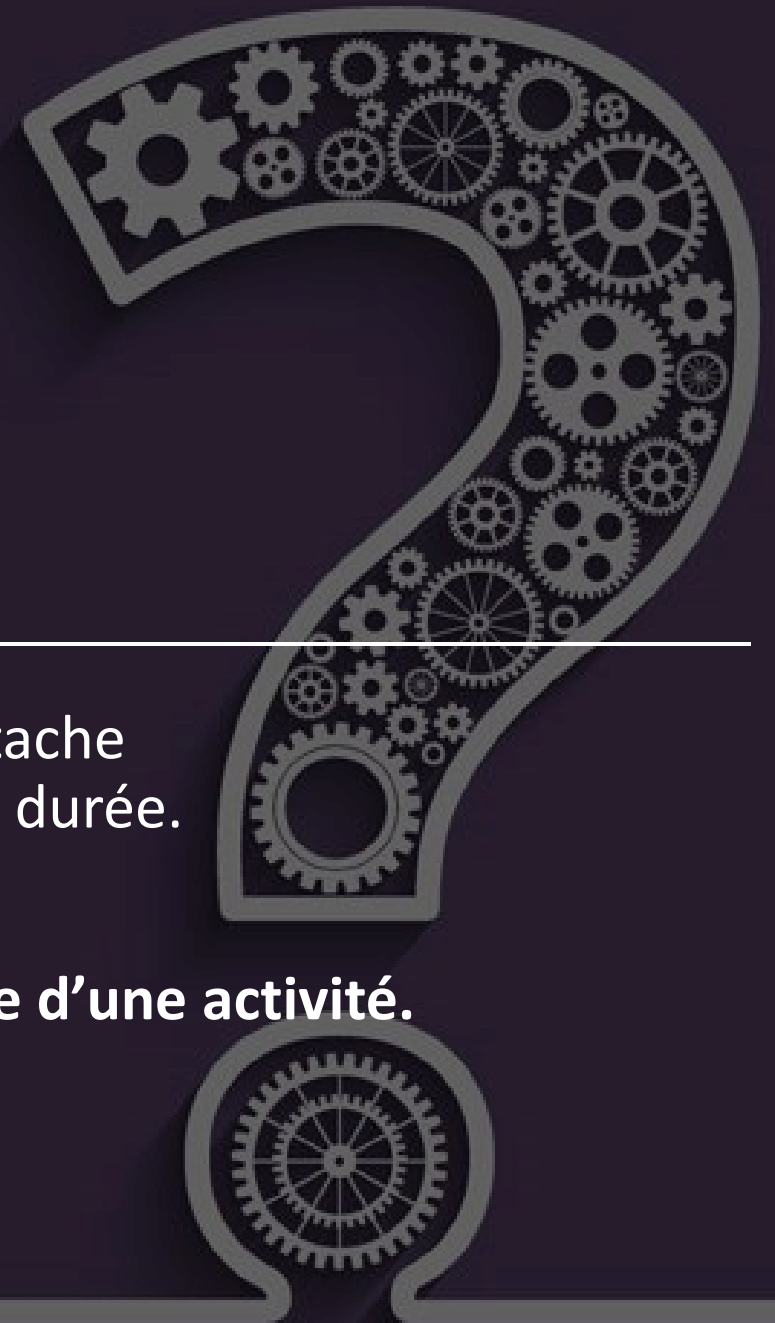
# Distinguez 2 techniques de virtualisation

## Réponse

**Machine virtuelle:** Abstraction complète de la machine. Interagit directement avec les **composants physiques de l'ordinateur**.

**Conteneur:** Contient toutes les dépendances nécessaires (services, bibliothèques...) pour exécuter une application. Utilise les fonctionnalités de l'OS hôte pour en simuler un autre.






---

Dans l'organisation de mes activités, je peux créer une tâche ou une séquence de tâches. Chaque tâche possède une durée.

**Proposer un design me permettant de calculer la durée d'une activité.**




Nommez 2  
contraintes sur le  
choix du langage  
de programmation

### Réponse

Le langage de programmation choisi pour un projet peut être contraint par de nombreux facteurs

1. **Spécification du contrat:** Le contrat peut spécifier un langage de programmation. *Le client pourrait avoir une dépendance à un langage dans son infrastructure.*
2. **Environnement cible:** Certains environnements (plateforme, OS) supportent un nombre restreint de langages. *Le web ne supporte que le JavaScript, HTML et CSS.*
3. **Expérience des développeurs:** L'équipe peut avoir des préférences liées à leur expérience. *L'équipe peut vouloir réutiliser des bibliothèques développées à l'interne ou familières.*





Quelle est la différence  
entre boîte noire et  
boîte blanche?

---

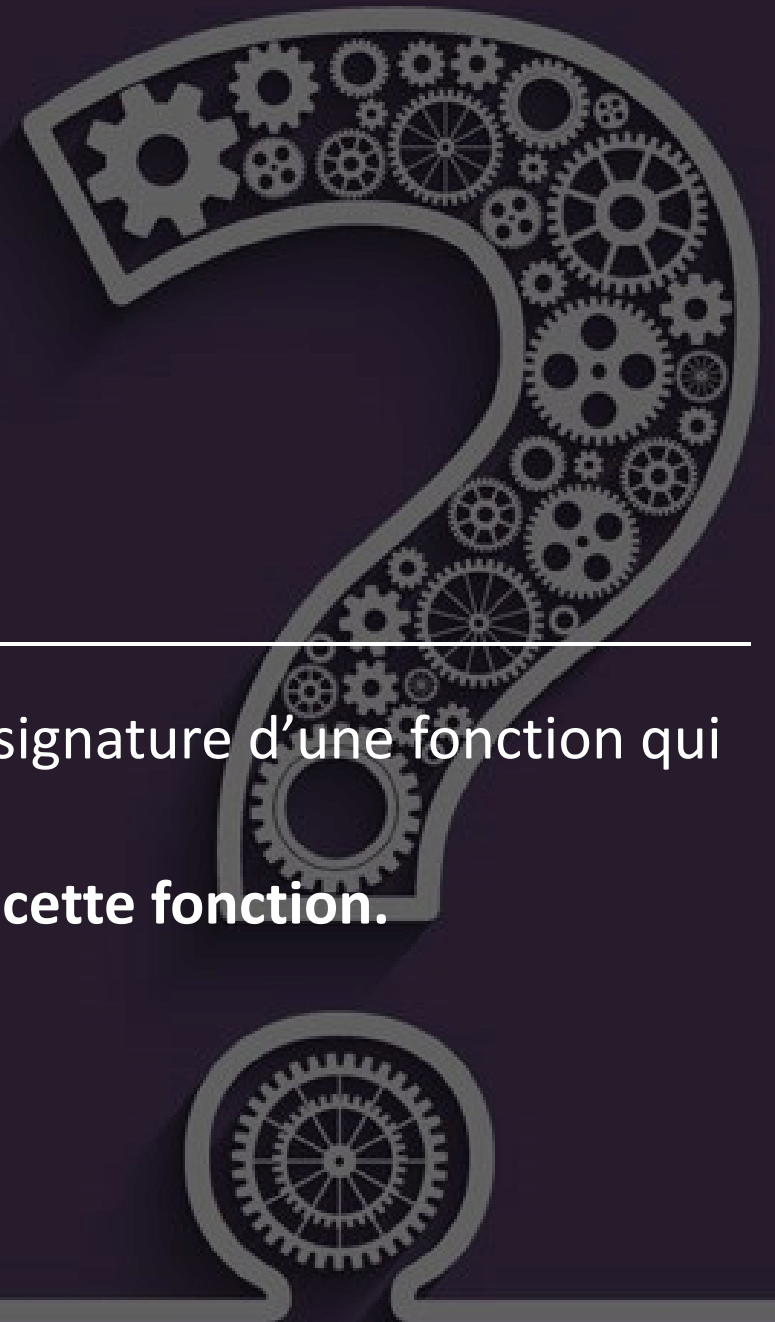
### Réponse

La **vérification** consiste à comparer la solution produite aux spécifications établies. On se demande ici si le logiciel a été fait correctement, indépendamment de l'usage réel qui en sera fait.

La **validation** consiste à évaluer la solution produite en fonction des besoins actuels des clients et utilisateurs. On se demande si le logiciel fait la bonne chose, pour répondre aux besoins.







---

**pop(int key, Map<int, Object> dict) : Object** est la signature d'une fonction qui enlève l'élément de dict indexé par key et le retourne.

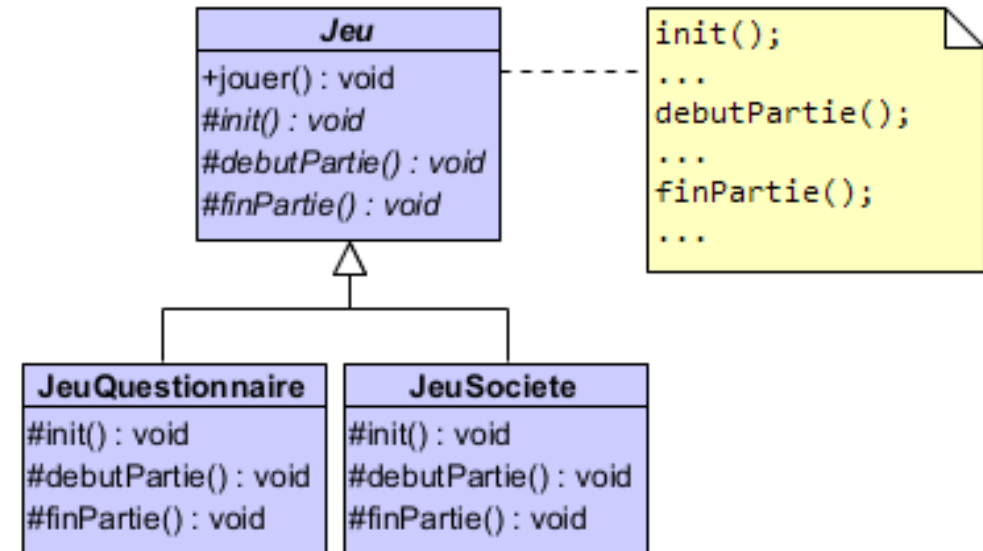
**Décrivez l'oracle de 3 cas de tests distincts pour tester cette fonction.**

Aucun code n'est exigé

# Oracle

key	dict	retour	Type de test
Peut importe	<code>== null</code>	Erreur	Échec: <code>NullPointerException</code>
Peut importe	<code>vide</code>	Erreur	Échec: Rien à afficher
Présent dans dict	<code>Non vide</code>	<code>dict[key]</code>	Succès
Absent dans dict	<code>Non vide</code>	Erreur	Échec: Clé n'existe pas

Décrivez le patron utilisé dans le diagramme de classe suivant



**Nom, But, Avantages**

**Réponse**

Ce diagramme représente le **patron de méthode** (*template method pattern*). Le but est de **moduler le comportement** d'un algorithme en définissant le **squelette** d'un algorithme dans une opération et en **reportant** certaines étapes à des sous-classes. Il permet **d'affiner** certaines étapes d'un algorithme sans changer sa structure.

Dessinez un diagramme de classe UML d'un régulateur de température automatisé d'une chambre intelligente

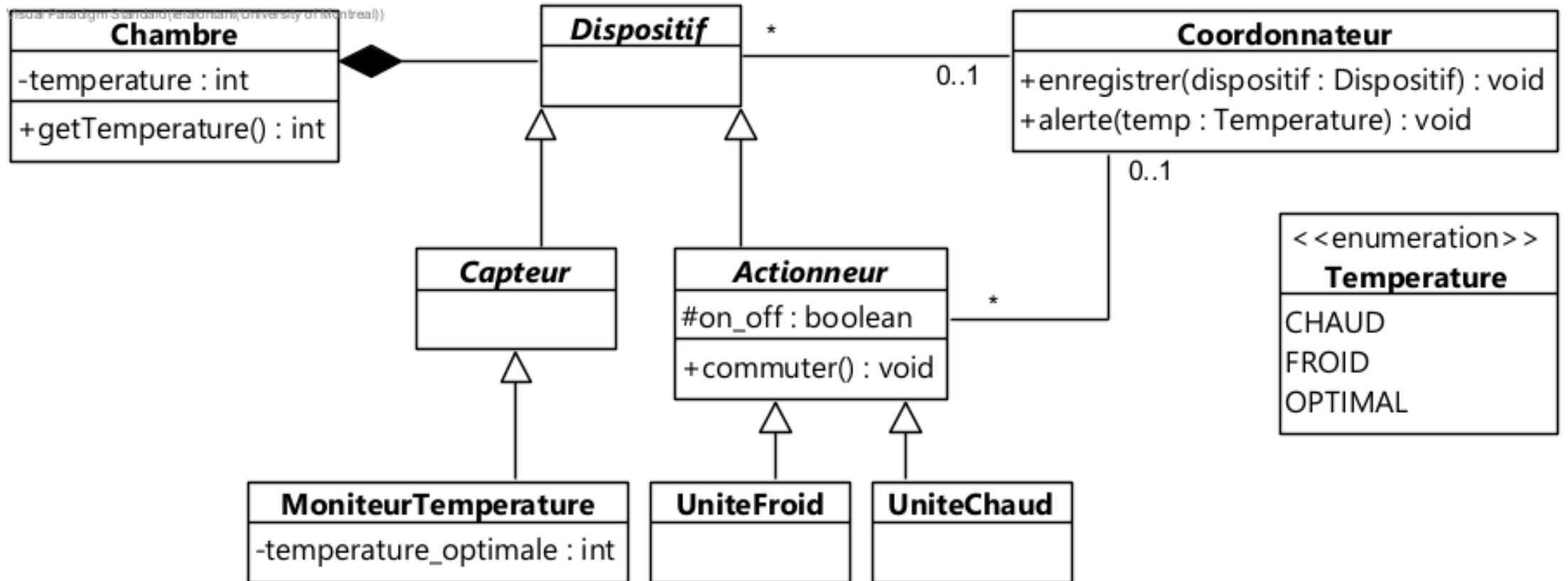


Une chambre contient deux sortes de dispositifs : des capteurs et des actionneurs. Un capteur détecte les événements de l'environnement et un actionneur réagit à ces événements.

Dans cette chambre, le seul capteur est un MoniteurTempérature, dans lequel (configuration) on indique la température optimale. Du côté des actionneurs, la chambre en possède deux: UnitéFroid et UnitéChaud. Tout actionneur a un mode *on\_off* qui peut être changé par l'action *commuter*. Le détails (comportement) de l'action dépend de l'actionneur.

Pour les réutiliser de façon modulaire, les dispositifs ne communiquent pas directement entre eux. Un Coordonnateur envoie et reçoit les messages qu'ils s'échangent. Un dispositif s'inscrit à au plus un coordonnateur en utilisant la méthode *enregistrer*. Le coordonnateur peut être averti d'un changement de température en utilisant la méthode *alerte* qui ne prend en paramètre que "CHAUD", "FROID" ou "OPTIMAL". Le coordonnateur peut aussi changer le mode d'un actionneur.

# Diagramme de classe



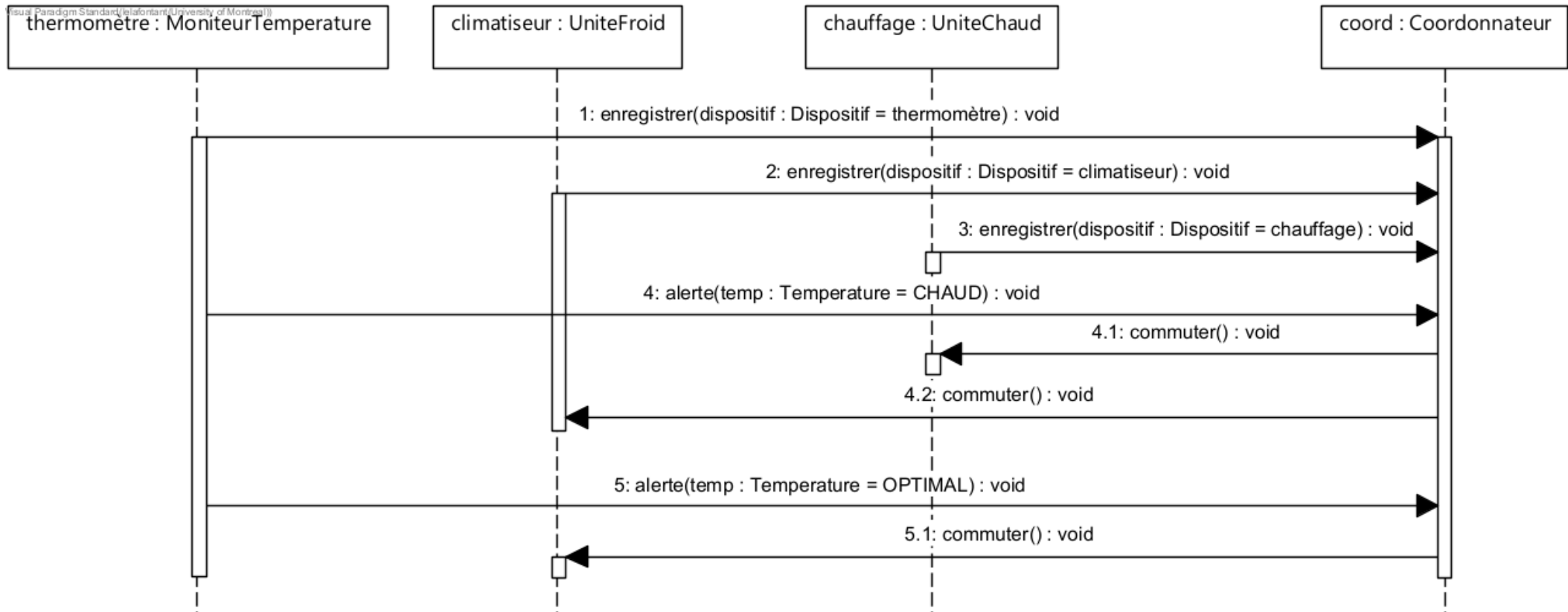
Dessinez un diagramme de séquence UML considérant les objets suivants: thermomètre (*MoniteurTempérature*), climatiseur (*UnitéFroid*) et chauffage (*UnitéChaud*).



D'abord, ils s'inscrivent tous à un objet de type *Coordonnateur* dans cet ordre. On suppose que le chauffage est allumé. Plus tard, un jour d'été très ensoleillé, le coordonnateur reçoit un message `alerte("CHAUD")` du thermomètre. À la réception de cette alerte, le coordonnateur éteint le chauffage et allume le climatiseur. Quand la chambre s'est refroidie et a atteint sa température optimale, le thermomètre notifie le coordonnateur qui éteint le climatiseur.

Votre diagramme doit être compatible avec le diagramme de classe précédent et représenter tous les objets nécessaires et les messages appropriés qu'ils s'échangent.

# Diagramme de séquence





# Café Hoocha Plateforme

---

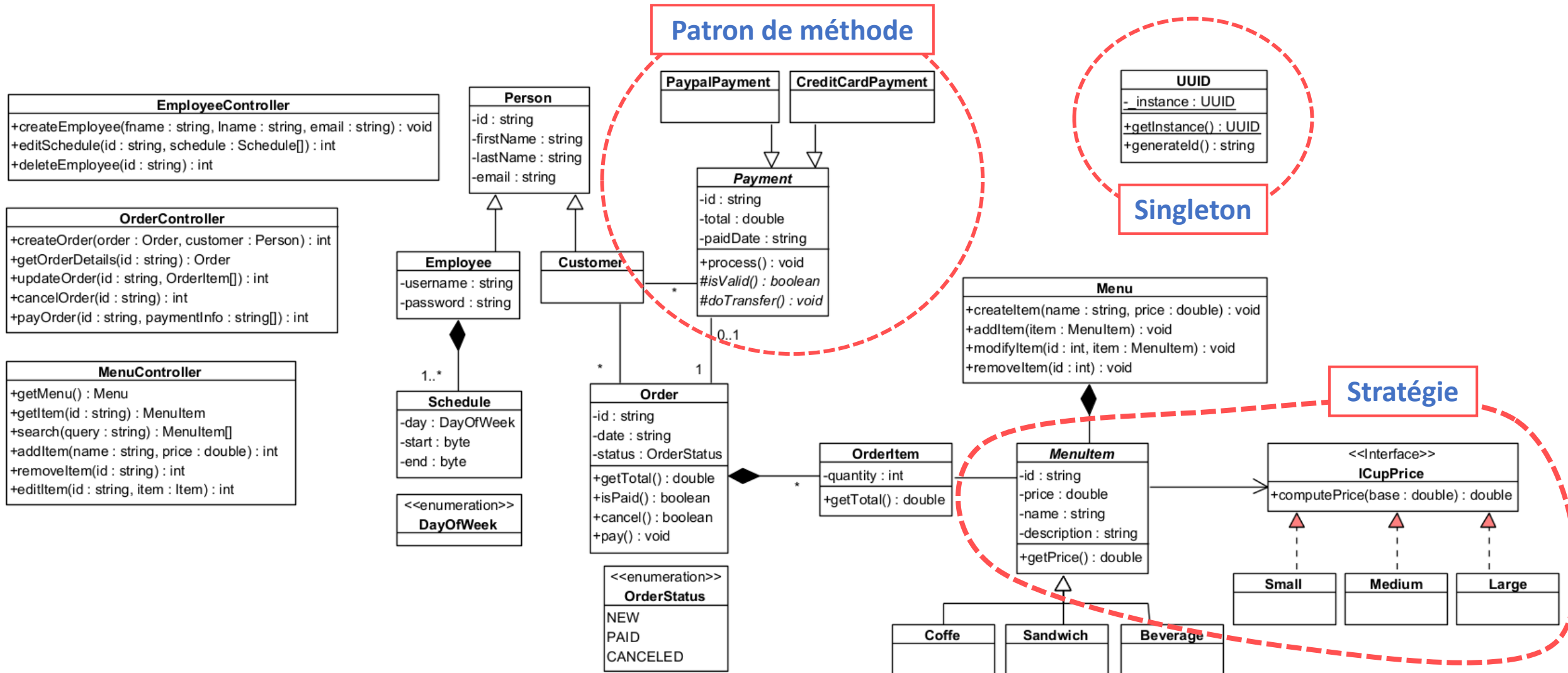
Le café Hoocha souhaite mettre sur pied une plateforme pour y inscrire son menu, gérer l'horaire du personnel et prendre les commandes de ces clients. La plateforme peut être utilisée par un client pour visualiser le menu et passer une commande; elle peut être utilisée par un membre du personnel pour modifier le menu, gérer les horaires et annuler une commande.

La plateforme supporte les paiements par carte de crédit ou Paypal. Dans les deux cas, le processus est le même: validation de la transaction suivie du transfert.

Le menu est composé de différents types de café, de boissons froides et de différents types de sandwich, croissants et muffins. Chaque item du menu a un prix et pour les cafés et boissons le prix dépend de la taille.



# Réponse possible (excluant la vue)





**KEEP  
CALM  
AND  
ACE YOUR  
EXAMS**