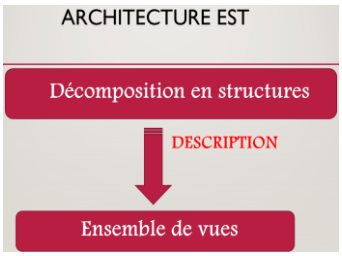
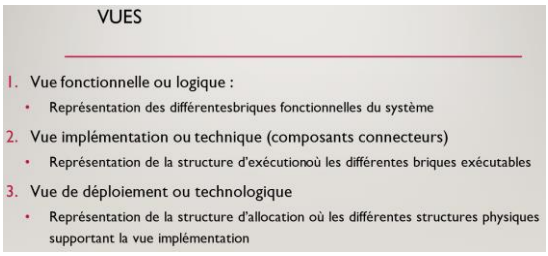
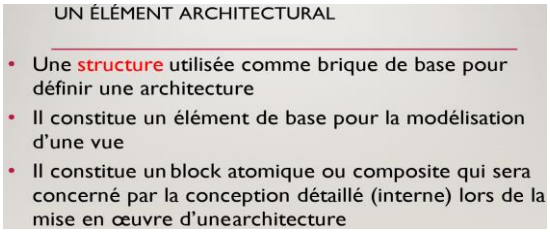
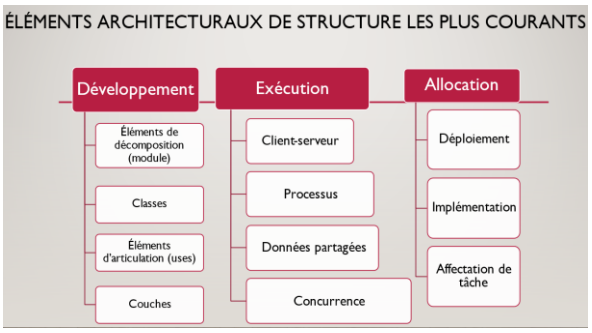
Objectifs : Manipuler les concepts de bases préliminaires des architecture logicielles

Rappels utiles et documentation :COURS **et** TD1









L'énoncé :

Une vidéothèque (magasin de location de films et jeux informatiques) décide de développer une application pour assister les employés du magasin dans leur tâches quotidiennes : le suivi et l'inventaire du stock (édition du catalogue), les clients, les articles loués et rendues, et les amendes accumulées par les clients. Les employés du magasin sont les seules qui auront accès à cette application qui sera nommée (VideoGameStore). Elle doit prendre en charge un ensemble de requêtes sur les clients et les articles en stock, ainsi qu'un ensemble de transactions qui modifient des informations des clients ou du stock.

Les requêtes	Les transactions
• Quels sont tous les films dans lesquels a joué un acteur donné ? • Y at-il un jeu ou un film avec un titre donné qui est disponible à la location • Quels sont tous les films qu'un client particulier a loués, avec leurs dates d'échéance ? • Quel est le solde du compte d'un client et quels sont tous les articles en retard ?	• Location et remise d'un article par un client • Ajout des amendes à son solde (pénalités de retard) • Ajouter un article au stock et mise à jour du catalogue • Ajouter un client

Deux modèles M1, M2 (voir les tableaux des pages suivantes) sont établis pour décrire 2 solutions conceptuelles possibles à cette application (VideoGameStore). Chaque modèle est décrit sommairement en termes de ses classes (leurs attributs sont supposés être privés, ayant des getters et setters mais seules les méthodes métiers qui sont publiques qui sont citées), les responsabilités de chaque classe, et comment les classes interagissent.

Les actions du travail demandé

1. Elaborer un DIAGRAMME DE CLASSE POUR CHAQUE MODELE (deux diagrammes)
2. Utilisez les diagrammes UML et donner les 3 vues de l'architecture de cette application selon le modèle M1 ensuite M2
3. Quelle est la différence entre les deux modèles M1 et M2 de point des contraintes non fonctionnelles.
4. Implémenter la solution du modèle M2 en JAVA.

Modalité de déroulement : Le TP doit être réalisé en binôme.

- 1 Séance de travail présentiel pour la compréhension du problème et les actions 1 et 2 (les vues du modèle M1.)
- Suite de l'action 2 et les actions 3 et 4 seront à remettre dans une semaine (avant la prochaine séance de TD/TP)

MODELE M1				
CLASSE	RESPONSABILITE	ATTRIBUTS	METHODES	
			NOM	TACHE
CheckedInGame	Un jeu en stock non actuellement loué	rentalPrice, title, itemID, platform.		
CheckedOutGame	Un jeu qui est actuellement en location	rentalPrice, title, itemID, platform, dueDate, clientLocateur (référence à un Objet Client associé à la personne qui loue le jeu en question)		
CheckedInMovie	Un film qui se trouve actuellement dans le stock	locationPrix, titre, ItemID, acteur		
CheckedOutMovie	Un film actuellement loué	rentalPrice, title, itemID, acteur, dueDate, clientLocateur (référence à un Objet Client associé à la personne qui loue le film en question)		
Client	Le compte client	accountBalance, name, customerID		
CommndProcessor	Le moteur de gestions des commandes, il prend en charge les requêtes	5 listes : • Jeux en stock • Jeux loués (chez les clients) • Film en stock • Film loués (chez les clients) • Clients du magasin	FindByTitle	Détermine si un article ayant un titre donné existe
			FindByActor	Retourne tous les films dans lesquels un acteur donné à joué
			IsCheckedOut	Détermine si un article est en location
			Solde	Retourne le solde d'un client donné
			OverdueItems	Retourne tous les articles qui sont actuellement en retard
			CheckOut	Location d'un article pour un client donnée à une date donnée
			CheckIn	Remise d'un article par un client donné à une date donnée
			AddCustomer	Ajoute un client à l'ensemble des clients du magasin
			AddStockItem	Ajoute un nouvel article au stock du magasin
STORE	Le noyau principal responsable de la création du CommandProcessor, en prenant les commandes utilisateur et les passant au CommandProcessor et affiche les résultats renvoyés	Un objet CommandProcessor Une IHM		

MODELE M2				
Classe	Responsabilité	Attributs	Méthodes	
			Nom	Tâche
StockItem	Superclasse de Film et Jeux	rentalPrice, title et itemID		
Client	accountBalance, nom, customerID.	Le client a des attributs accountBalance, name et customerID.		
Film	Encapsule l'article film	Hérite de StockItem ajoute un attribut acteur		
Jeux	Encapsule l'article Jeux	Hérite de StockItem ajoute un attribut plateforme (indiquant la plate-forme d'exécution du jeu)		
RentedItem	Représente un article qui est loué par un client	customerID, itemID, dueDate		
Map <k, T>	Représente une collection d'objets de type T, chaque objet associé à une clé de type k.		Find (k)	Retourne l'objet de type T associé à la clé k,
			Insert (k, T)	Insère un objet T dans la MAP associée à une clé k
			Delete (k),	supprime l'objet de type T associé à la clé k.
			Iterator ()	Iterator (), qui retourne un MapIterator <k, T>
MapIterator <k, T>	Une généralisation d'une référence à la mémoire		Begin ()	Définit l'itérateur sur le premier élément de la Map <k, T>
			Get ()	Retourne un élément de type T à l'emplacement actuel de l'itérateur,
			HasNext ()	Renvoie un booléen s'il reste des éléments dans la carte pour itérer,
			Next ()	Avancer l'itérateur vers l'élément suivant dans la séquence itérée.
QueryProcessor	Gère les requêtes de l'utilisateur qui n'entraînent pas de modifications des données client ou stock	Réf Map <String, Customer> Réf Map<String, StockItem> Réf liste RentedItem	FindByTitle()	Détermine si un article avec un titre donné existe
			NdByActor()	Renvoie tous les films dans lesquels un acteur donné joue
			IsCheckedOut	Détermine si un article est en location
			Solde()	Retourne le solde d'un client donné
			OverdueItems( )	Retourne tous les article actuellement en retard
TransactionProcessor	Gère les commandes de l'utilisateur qui entraînent des modifications dans le client ou le stock	Réf Map <String, Customer> Réf Map<String, StockItem> Réf liste RentedItem	CheckOut()	Louer un article à un client à une date donnée
			CheckIn()	Retour d'un article par un client donné à une date donnée
			AddCustomer( )	Ajout d'un client au clients du magsin
			AddStockItem( )	Ajoute d'un nouvel article au stock du magasin
STORE	Noyau principal responsable de la création de maps des clients et des articles en stock, de transfert des commandes de l'utilisateur processeur de requête (QueryProcessor) ou de transaction (TransactionProcessor) et l'affichage des résultats			