Conception Orientée Objet Examen 2023-2024

Durée : 3 heures - Documents autorisés sauf livres et appareils électroniques - Utilisation du crayon autorisée

Exercice 1. Diagramme d'activité : cabine photographique (3,5 points)

Considérons une cabine photographique automatique composée de plusieurs modules : « IHM », « caméra », « paiement » et « production ». Au début, le client doit choisir sur « IHM » le type de photo qu'il souhaite prendre : photo d'identité ou photo autoportrait. Pour les photos autoportrait, le client a besoin de spécifier le style (effets spéciaux) à appliquer sur les photos. Pour les photos d'identité, le client n'a rien à spécifier et « IHM » diffuse tout simplement une petite vidéo expliquant comment prendre des photos d'identité conformes. Après les étapes mentionnées ci-dessus, le module « paiement » demande au client de régler le montant dû en fonction du type de photo choisi. Il n'y a qu'un seul mode de paiement : par espèce et la machine ne rend pas la monnaie… Après le paiement, le module « caméra » se met au travail en proposant au client au maximum trois essais pour prendre sa photo. A chaque essai, le client prend la photo, la visualise sur « IHM » et s'il en est satisfait, il en informe le module « caméra », sinon, il peut continuer un autre essai. Dans tous les cas, le client doit choisir une photo à la fin. Enfin, le module « production » termine le travail : pour les photos autoportrait, il imprime directement les photos et les fournit au client ; pour les photos d'identités, il les télécharge sur Internet et fournit au client un code de téléchargement. Pendant toute la procédure, l'annulation de la part du client n'est possible qu'avant l'étape de paiement. Modélisez les activités entre le client et différents modules de la cabine photographique par un diagramme d'activité UML.

Exercice 2. Diagramme d'état-transition : robot arroseur (4 points)

On souhaite modéliser un robot qui peut automatiquement arroser les plantes dans un environnement donné. Une fois allumé, le robot s'initialise et puis il commence à capturer, avec ses caméras, les images de l'environnement. Ensuite, il traite les images capturées afin de trouver les plantes à arroser. Après certains calculs basés sur les données issues du traitement d'images, il se dirigera vers la plante la plus proche. A l'arrivée devant la plante, il vérifiera tout de même, grâce à une base de données à distance, si la plante doit être arrosée. On ne devrait pas arroser une plante plus d'une fois par jour ! Si oui, il arrose la plante. Après l'arrosage de la plante (ou pas besoin d'arroser la plante), le robot se remet en route pour une nouvelle aventure destinée à la prochaine plante en répétant les mêmes actions (capturer, traiter, calculer, etc.). Bien sûr, le robot ne traite chaque plante qu'une seule fois. Quand le robot n'arrive plus à trouver de plantes non traitées dans l'environnement, il s'éteint tout seul. Au cours de toutes les actions du robot, on peut toujours l'éteindre manuellement quand on veut. En particulier, quand la pile du robot devient presque vide, le robot se met en « pause » automatiquement en émettant un signal sonore. Après le changement (rapide) de pile, le robot peut reprendre l'action en cours. Ne vous inquiétez pas, pendant le changement (rapide) de pile, le robot ne s'éteint pas grâce à une mini batterie de secours qui se charge automatiquement à chaque fois que le robot est en déplacement entre les plantes. Modéliser ce robot par un diagramme d'état-transition UML.

Exercice 3. Domain Diven Design : réseau ferroviaire (3,5 points)

On souhaite avoir un programme Java permettant de gérer un réseau ferroviaire d'une grande métropole. Le réseau est composé de lignes de métro, de lignes de RER et de lignes de train de banlieue. On distingue les différents types d'arrêts (station, gare, gare Grande Ligne) desservis par ces différents types de train. On s'intéresse à la création, modification et suppression des lignes et des arrêts. On a besoin de gérer les horaires de train de différentes lignes. On a aussi besoin de lier les différents types d'arrêts afin de créer les correspondances entre les lignes. On ne gère **pas** les trains de Grande Ligne qui partent des gares Grande Ligne vers d'autres villes (métropoles).

Langage ubiquitaire :

Métro : trains légers qui circulent entre différentes stations à une vitesse moyenne
Station : une ligne de métro est composée de différentes stations qui ont une capacité d'accueil moyenne.
RER : trains avec une capacité de voyageurs grande qui circulent entre différentes gares à une vitesse élevée
Gare : une ligne de RER est composée de différentes gares qui ont une capacité d'accueil élevée.
Train de banlieue : trains avec une capacité de voyageurs grande qui partent uniquement des gares Grande Ligne et qui circulent entre différentes gares à une vitesse élevée
Gare Grande Ligne : ces gares ont une capacité d'accueil très élevée et sont le point de départ des trains de banlieue.
Horaire d'un train : un ensemble d'arrêts desservis par le train et on indique une heure d'arrivée du train pour chaque arrêt.

Modélisez ce problème avec un diagramme de classe UML en suivant l'approche DDD. Dans le diagramme, vous indiquerez les noms des classes*, ainsi que les liens entre les classes*. Vous légenderez / indiquerez aussi les agrégats avec la classe* racine, les types de classes* « Entité » et « Objet de valeur », ainsi que les classes* de type service : services fonctionnels, factory et repository. Vous n'avez **pas** besoin d'indiquer les attributs ni les méthodes dans les classes.

* Les classes peuvent être classes concrètes, classes abstraites ou interfaces.

Exercice 4. Principes d'objet et design patterns (5,5 points)

5.1 Pourquoi le pattern « *Iterator* » respecte le principe d'inversion de dépendance (*DPI*). (1 point)

5.2 Comparez le pattern « *Adapter* » et le pattern « *Decorator* ». (1 point)

5.3 Considérons un mini système de produits commercialisés. Il y a deux types de produit : type pour utilisation personnelle et type pour utilisation professionnelle. Le produit **contient** une pièce spéciale importante. Cette pièce a deux modèles : pièce intégrée ou pièce détachée. La fabrication de ces deux modèles de pièce doit être gérée avec une solution extensible et maintenable : 1) on traite séparément les différentes fabrications de différents modèles de pièce 2) on doit pouvoir ajouter facilement un nouveau modèle de pièce et traiter sa fabrication sans modifier le système existant. Et ces sont les mêmes exigences pour la fabrication des deux types de produit. De plus, pour l'utilisateur de ce système de produits, l'interaction avec le système doit être simplifiée. C'est-à-dire que l'utilisateur n'a pas forcément besoin de connaître tous les détails de la fabrication et des opérations des produits / pièces. Modéliser ce problème par un diagramme de classe UML. Vous y indiquerez **uniquement** les noms des classes ainsi que les relations entre les classes, pas besoin de mentionner les attribut, ni les méthodes. Indiquez les **design patterns** utilisés dans votre solution proposée. (3,5 points)

!!!Attention : Vous choisirez UN SEUL exercice à faire parmi 5a et 5b

Exercice 5a. Diagramme de cas d'utilisation : cuisine automatisée (3,5 points)

Considérons une cuisine automatisée capable de gérer la tâche d'achat des ingrédients et celle de cuisson. Pour l'achat des ingrédients, le système dispose de deux modes de fonctionnement : mode automatique qui consiste à acheter les ingrédients conseillés selon l'état de santé des membres de la famille ; mode manuel qui consiste à acheter les ingrédients choisis manuellement par son maître (membres adultes de famille). Pour les deux modes de fonctionnement, la cuisine automatisée fait l'achat en ligne. Et le maître n'a qu'à recevoir les livraisons et mettre les ingrédients reçus dans le stockage (frigo) de la cuisine. Pour le côté de cuisson, on souhaite pouvoir programmer les repas de la semaine (indiquer les recettes souhaitées pour chaque repas). La cuisine peut aussi proposer automatiquement les repas pour la semaine selon les ingrédients en stockage et l'état de santé des membres la famille. Par conséquent, il est nécessaire que le maître puisse renseigner les informations sur l'état de santé de tous les membres de la famille. La cuisson automatique peut être démarrée uniquement par le maître. Pour les enfants, ils n'ont accès à qu'une seule fonctionnalité supplémentaire : prendre quelque chose pour grignoter (avec une quantité limitée). Les adultes peuvent également grignoter mais avec une quantité illimitée. Modéliser cette cuisine automatisée avec un diagramme de cas d'utilisation UML.

Exercice 5b. Diagramme de séquence : véhicule de transport (3,5 points)

Considérons un véhicule autonome qui permet de transporter automatiquement des marchandises d'un lieu à un autre. Le véhicule est équipé de plusieurs modules : un chargeur, un GPS, un radar, une caméra et un moteur intelligent qui dirige toutes les actions des autres modules. Avant le départ, le moteur demande au chargeur de charger les marchandises et calcule l'itinéraire optimal pour la destination grâce au GPS. Et puis, il se met en route. Sur la route, la caméra capture sans arrêt des images en 360° autour du véhicule et le radar détecte en même temps les autres véhicules et d'éventuels obstacles. Le moteur réagit sans arrêt avec les données transmises par la caméra et le radar, afin de garantir un trajet en sécurité. Pendant le trajet, le moteur peut éventuellement modifier l'itinéraire grâce au calcul en temps réel du GPS, en prenant en compte l'évolution du trafic indiqué par le GPS. Le moteur émet un message (on ignore le récepteur du message) à l'arrivée à la destination. A la destination, le moteur décharge les marchandises grâce au chargeur. Modéliser les interactions entre les différents modules de ce véhicule par un diagramme de séquence UML.