Examen Final GL52

P2017
Durée: 2h
Aucun document autorisé
L'exercice 1 sur une copie à part
Le barème est donné à titre indicatif (±1)
Mercredi 28 juin 2017

Remarque : les diagrammes UML doivent être complets et clairement présentés.

1 Exercice 1 : Application de gestion des Rapports Quotidiens de Vol pour la police nationale. (10 points, à faire sur 1 copie séparée)

On souhaite réaliser une application web pour la gestion des Rapports Quotidiens de Vol (RQV) de véhicules pour la police nationale.

On distingue initialement deux types d'utilisateurs pour ce système : les victimes et les témoins. Chacun de ces utilisateurs peut créer une déclaration de vol, en y indiquant son rôle (victime, témoin ou bien les deux), ses informations personnelles (son numéro de sécurité sociale, nom, prénom, adresse, tél), le type de véhicule volé (véhicule à moteur ou bien bicyclette) ainsi que les différentes informations disponibles qui l'identifient (couleur, marque, numéro de série pour les bicyclettes, immatriculation pour les véhicules à moteur, description générale), la date, l'heure et le lieu (avec tous les détails disponibles : numéro de la rue, ville, code postal,...) du vol.

Le système attribue à chaque déclaration un identifiant unique, que l'utilisateur peut utiliser pour pouvoir éditer la déclaration (ajouter des informations, supprimer la déclaration), avant de sauvegarder la déclaration. Le système enregistre, pour chaque déclaration, la date de sa dernière modification.

On distingue également un autre type d'utilisateurs : l'agent policier qui se charge de la création des Rapports Quotidiens de Vol. Un RQV est relatif à une date particulière, il contient toutes les déclarations de vols effectuées dans ce jour, la liste de celles qui ont été modifiées et de celles qui ont été résolues. Lorsqu'un véhicule déclaré volé est retrouvé, l'agent policier modifie l'état de la déclaration concernée. Evidemment, l'agent policier doit s'authentifier pour pouvoir accéder à cette application.

Question 1 (4 points) Fournir le diagramme UML de classes (classes, attributs, associations, etc. mais les méthodes ne sont pas nécessaires, elles seront décrites dans le diagramme de séquence de la question 2) qui décrit la structure d'une solution logicielle permettant d'implanter l'application web décrite ci-dessus.

Question 2 (3 points) Fournir le diagramme UML de séquence « niveau objet » détaillé du processus permettant à un policier de modifier l'état d'une déclaration lorsque le véchiule concerné a été retrouvé. On suppose que la séquence démarre avec un utilisateur visiteur. Cette description doit être cohérente avec l'architecture proposée à la question 1.

Question 3 (3 points) Proposer un diagramme UML états-transitions pour modéliser la dynamique d'états d'une déclaration de vol. Cette description doit être cohérente avec l'architecture proposée à la question 1 et la séquence de la question 2.

2 Exercice 2 : Méthode B (5 points)

Soit la machine Theatre qui gère les réservations de places pour une pièce de théâtre.

MACHINE Theatre
SETS
$SIEGES = 1..200$ (les sièges vont de 1 à 200)
VARIABLES
$occupes, nb\_libre$ (occupes : les places occupées, nb_libre : nombres de places libres)
INVARIANT
$occupes \subseteq SIEGES \wedge nb\_libre \in 0..200 \wedge nb\_libre = 200 - card(occupes)$
INITIALIZATION
$occupes, nb\_libre := \varnothing, 200$ ($\varnothing$ représente l'ensemble vide)
OPERATIONS
Reserver $\widehat{=}$
PRE $nb\_libre \neq 0$
THEN
ANY pp WHERE $pp \in SIEGES - occupes$ THEN
$occupes, nb\_libre := occupes \cup \{pp\}, nb\_libre - 1$
END ;
END

Question 1 (5 points) Donner la démonstration détaillée (expliciter l'application des substitutions) de la preuve de cohérence de la machine Theatre.

3 Exercice 3 : Spécification algébrique (5 points)

On considère le type abstrait Liste, permettant de construire et de manipuler des listes dont le type des éléments noté ELT est quelconque. Les opérations associées au type Liste sont :

init : crée une liste vide,

ajouterTete : ajoute un élément en tête d'une liste,

ajouterDer : ajoute un élément en queue d'une liste,

supprimerTete : supprime l'élément de tête d'une liste

supprimerDer : supprime l'élément en queue d'une liste

appartient : teste l'appartenance d'un élément à une liste,

Question 1 (5 points) En considérant les opérations init, ajouterTete comme constructeurs primitifs, donner la spécification algébrique du type abstrait Liste dont le type des éléments (noté ELT) est quelconque.