Modélisation et programmation par objets 2

Tous documents sur support papier autorisés. Durée : 2h

1 Interface, classe générique, streams, assertion (environ 2/3 des points)

Dans cette partie, nous développons des éléments pour la représentation de halles thématiques dans un parc exposition. Une halle thématique comprendra plusieurs stands qui partagent une même thématique liée à l'exposition.

Question 1. Ecrivez en Java une interface représentant un stand dans le contexte d'une halle d'un parc d'exposition. Pour un stand, on peut connaître (1) un descriptif, (2) une durée (en heures), (3) une surface (en m^2), (4) un prix de location de base par heure et par m^2, qui est le même pour tous les stands de l'exposition, est supposé constant et de visibilité `public` (5) un prix de location calculé comme le produit de la surface par la durée.

Question 2. Ecrivez en Java une classe concrète représentant un stand et implémentant l'interface de la question précédente. N'écrivez que les attributs et que les méthodes suffisant à rendre la classe concrète. N'écrivez pas les constructeurs mais ils seront supposés exister, de même que les accesseurs non écrits.

Question 3. Ecrivez en Java une assertion vérifiant que le prix de location calculé est bien positif ou nul. Expliquez et indiquez avec précision à quel endroit vous mettez cette assertion.

Question 4. Une halle thématique possède un nom, une surface et une liste de stands. Ecrivez en Java l'entête et les attributs (avec initialisation) d'une classe générique représentant une halle thématique. La classe générique est paramétrée par un type de stands (conformes à l'interface définie plus haut).

Question 5. Ecrivez en Java, dans la classe représentant les parcs exposition, une méthode `prixTotalLocation-StandSurfaceSupÀ` retournant la somme des prix de location des stands dont la surface est supérieure à une certaine surface passée en paramètre. Cette méthode doit être impérativement écrite en utilisant les **opérations sur les flots (streams) et les lambdas expressions** et non pas une classique itération avec un `for` ou un `while`, ni avec un itérateur.

Question 6. Dessiner en UML un diagramme de classes correspondant aux éléments écrits en Java dans les questions 1 à 4.

2 Diagramme statique et diagramme dynamique (environ 1/3 des points)

Question 7. Vous représenterez en UML la situation suivante par (1) **un diagramme de classes** et (2) **un diagramme de séquences** cohérents l'un avec l'autre. Nous suggérons de commencer par réaliser le diagramme de séquences pour identifier les opérations nécessaires sur les différentes classes. Nous rappelons que le diagramme de séquence représente des échanges de messages entre instances, le long d'une ligne de vie ; des méthodes doivent être prévues dans les classes pour étiqueter ces messages.

On s'intéresse, dans le domaine bancaire, à la description (ici simplifiée) d'une procédure de virement en ligne par un client sur le compte d'un autre client.

Le premier `client` envoie une requête de virement par l'interface en ligne de sa `banque-locale`, cette requête comprend la somme à virer et le numéro de `compte` du second client.

Si le solde du compte du premier client ne permet pas ce virement, un message d'erreur lui est retourné et l'opération s'arrête.

Sinon la banque du premier client envoie une requête à la `banque-centrale` pour vérifier que le numéro de compte du second client existe et récupérer sa référence.

Si le numéro de compte du second client n'existe pas, un message d'erreur est retourné par la banque centrale à la banque locale qui le fait suivre au premier client et l'opération s'arrête.

Sinon, la référence du compte du second client est transmise à la banque locale et la banque locale effectue le virement : elle retire la somme du compte du premier client et l'ajoute au compte du second client.

Le premier client et la banque locale du second client reçoivent une notification de la part de la banque locale du premier client. La banque locale du second client fait suivre ce message à ce dernier.