Département Informatique

Examen POO de ISIL A et ISIL B

2015/2016

Veuillez lire tout l'énoncé avant de traiter le sujet

Exercice

On s'intéresse à une entreprise qui gère un ensemble de parkings de véhicules, un parking gère deux types de clients, les clients abonnés et les clients non abonnés (libres).

- Un client abonné est caractérisé par le nom, prénom, adresse, matricule du véhicule (long), la date d'abonnement, le nombre d'heures de stationnement autorisées, la date et l'heure de sa dernière entrée au parking.
- Un client non abonné est caractérisé par le matricule du véhicule, la date et l'heure d'entrée et le numéro de place de stationnement.

1. En considérant les classes Date et Heure comme prédéfinies :
- Implémenter les classes nécessaires en évitant les redondances.
- Ajouter pour chaque classe un constructeur et redéfinir la méthode toString

Indication : A la création d'un objet de type client abonné les informations suivantes sont données : son nom, prénom, matricule, sa date d'abonnement et le nombre d'heures de stationnent autorisées.

A la création d'un objet de type client non abonné, son matricule, la date et heure d'arrivée et le numéro de place affecté sont précisés.

- Redéfinir la méthode equals pour les clients abonnés : Deux abonnés sont considérés égaux si leurs nombres d'heures autorisées de stationnement sont égaux.

2. Définir une classe Parking caractérisée par un nom, un code séquentiel automatique, une adresse, le nombre de places réservées pour les clients non abonnés (nbPlaceNonAb), le nombre de places réservées pour les abonnés (nbPlaceAb),le prix d'une heure de stationnement pour les clients non abonnés (prixPlaceNonAb), le prix d'une heure de stationnement pour les clients abonnés (prixPlaceAb), une liste placesLibres de type ArrayList qui contient les numéros de places libres des clients non abonnés.
La classe Parking est caractérisée aussi par une table listClients de type hashMap des clients dont le matricule est pris comme clé.
- Définir le constructeur de cette classe de telle sorte à remplir la liste placesLibres par tous les numéros de places libres de 1 à nbPlaceNonAb.

Ajouter les méthodes suivantes

- Abonné estAbonné (long mat) qui vérifie si le client de matricule mat est un client abonné du parking. Dans ce cas, elle retourne l'objet correspondant sinon l'objet null est retourné.

- int nbPlacesLibresAbonnés() qui retourne le nombre de places libres des clients abonnés.

- int numLibre() qui retourne un numéro de place libre qu'on peut affecter à un client non abonné.

- void arrivéeClient (long mat, Date d , Heure h) :

Si le client est un abonné, elle vérifie si le nombre de ses heures de stationnement autorisées est supérieur à zéro. Dans ce cas, les informations sont enregistrées sinon un message d'erreur est affiché indiquant au client qu'il doit renouveler son abonnement.

Par contre si le client est libre (non abonné), elle vérifie la disponibilité des places libres. S'il existe une place libre, ce client est inséré dans la liste listClients.

- void sortie (long mat, Date d, Heure h) :

Si le client correspondant est un abonné, son nombre d'heures autorisées de stationnement est mis à jour en lui retranchant le nombre d'heures de stationnement (calculé par la différence entre la date et l'heure de sortie et la date et l'heure de stationnement). Si le nombre d'heures autorisées est négatif une exception est déclenchée (jetée) en affichant le nombre d'heures à payer. On suppose que la méthode nbHeures qui réalise la différence est une méthode statique de la classe Date (on ne vous demande pas de l'écrire), cette méthode retourne le nombre d'heures.

La sortie d'un client non abonné calcule et affiche une facture contenant les informations du client, sa date et heure de sortie, le nombre d'heures de stationnement et la somme à payer, cette méthode doit aussi mettre à jour les listes placesLibres et listClients.

Indication :

La méthode void sortie (long mat, Date d, Heure h) doit appeler la méthode void sortie (Date d, Heure h) du client correspondant.

4. La classe EntrepriseParking contient la liste des parkings listParking de type ArrayList. Ajouter le code qui permet de trier cette liste par rapport au nombre de places libres des non abonnées. Mettre à jours les classes définies dans les questions précédentes si nécessaire.

Indications :

La méthode **put** de HashMap insère le couple clé, valeur.
La méthode **get** de HashMap retourne la valeur associé à la clé donnée en entré.
La méthode **values** de HashMap retourne toutes ses valeurs.
L'opérateur **instanceof** permet de tester si un objet est une instance d'une classe donnée.
La méthode statique **sort** de Collections permet de trier une liste par ordre croissant.