TP 2 (partie 2) : création de classes

Exercice I : Agence de location

Cet exercice concerne la gestion (très simplifiée) d'agences de location d'automobiles. Chaque agence gère un groupe de clients, un groupe de voitures et ses locations.

Question 1
Créez la classe Voiture. Une voiture se caractérise par son immatriculation (String), son modèle (String), son nombre de kilomètres parcourus (int) et son tarif de location au kilomètre (float). Écrivez un constructeur qui demande la saisie au clavier des caractéristiques de la voiture créée. Écrivez des méthodes d'instance pour accéder aux variables d'instance et une méthode d'instance pour l'affichage.

La question suivante permet de tester cette classe.

Question 2
Une instance d'ArrayList est un tableau dynamique. Dans le *main* de la classe Voiture, déclarez et utilisez une instance d'ArrayList de 10 voitures. Insérez quelques voitures, affichez le contenu de l'ArrayList, supprimez quelques voitures et affichez le contenu de l'ArrayList.

Question 3
Créez la classe Client. Un client se caractérise par un numéro (int généré automatiquement lors de chaque création), d'un nom (String) et d'un domicile (String). Réfléchissez au moyen de générer automatiquement le numéro des clients. Ajoutez une méthode pour afficher les informations d'un client.
Testez cette classe par la création de deux clients et leur affichage.

Question 4
Proposez une définition pour la classe Date. Une date se caractérise par un jour (int), un mois (int) et une année (int). Le constructeur demande les informations au clavier, et on souhaite pouvoir afficher une instance de Date. Testez cette classe en créant et en affichant plusieurs dates, puis en modifiant une date et en la réaffichant.

Question 5
Proposez une définition pour la classe Location. Une location se caractérise par un numéro (int) généré automatiquement, une voiture (Voiture), un client (Client), une date de location (Date) et une date de retour (Date) et un nombre de kilomètres. Le constructeur admet la voiture, le client et la date de location comme paramètres. La

date de retour est mise à la valeur null et le nombre de kilomètres parcourus (inconnu à la création) est initialisé avec le kilométrage initial de la voiture louée. Une méthode pour l'affichage doit être prévue (bien distinguer le cas où la date de retour est null ou non, car dans le premier cas il s'agit d'une location en cours, et d'une location terminée dans le deuxième cas).
Testez cette classe par la création d'une location, son affichage, la modification de sa date de retour puis à nouveau son affichage.

Question 6
Proposez une définition pour la classe Agence. Une agence est caractérisée par son nom et possède un parc de voitures, une clientèle et une liste de contrats (locations). Les voitures, les clients et les locations sont conservés dans trois instances d'ArrayList distincts (de capacité initiale 100). Proposez un constructeur qui initialise l'agence avec quelques voitures, clients et locations en cours. Ce constructeur ne devra pas faire de lecture au clavier, vous serez donc obligé de définir de nouveaux constructeurs pour les classes Voiture, Date et Client. Des méthodes pour l'affichage de la liste des voitures, de la liste des clients et de la liste des locations doivent être prévues.

Question 7
Proposez et testez une méthode pour enregistrer une location (on doit demander le numéro du client, vérifier s'il est bien enregistré, lui proposer les véhicules disponibles et enregistrer sa location). Attention, définir cette méthode peut demander de rajouter d'autres méthodes.
Pour programmer cette méthode, vous vous baserez sur cet algorithme :

```
Saisir n° client
TantQue non existe(1)
        signaler erreur
        saisir n° client
finTantQue
afficher la liste des voitures disponibles (2)
saisir immat d'une voiture
TantQue non existe ou non dispo
        signaler erreur
        afficher la liste des voitures disponibles
        saisir immat
finTantQue
saisir la date de départ
créer la location
l'enregistrer dans la liste des locations
```

(1) on pourrait aussi envisager la création d'un nouveau client.

(2) pour compléter, on pourra envisager le cas où le nombre de voitures disponibles est égal à 0, auquel cas la méthode se termine en affichant un message adéquat.

Question 8
Proposez et testez une méthode pour enregistrer le retour d'une location. Attention, définir cette méthode peut demander de rajouter d'autres méthodes.
Pour programmer cette méthode, vous vous baserez sur cet algorithme :

```
saisir immat
TantQue non existe OU pas en cours de location
        signaler erreur
        saisir immat
finTantQue
afficher les caractéristiques de la location
saisir date du jour (1)
saisir kilometrage retour
TantQue km retour<= km depart
        signaler erreur
        saisir kilometrage retour
finTantQue
calculer la distance parcourue
calculer et afficher le prix à payer
modifier la location (dateRetour et distance)
modifier aussi la voiture (son kilométrage)
```

(1) pour compléter, on pourra ajouter un test sur la date de retour : elle doit être postérieure ou égale à la date de location. Il faudra donc ajouter à votre classe Date une méthode de comparaison pour comparer deux dates.

Exercice II : Ensembles

Nous souhaiterions modéliser un ensemble d'entiers sous la forme d'objets de la classe d'ArrayList.

Par définition, cet ensemble ne contiendra que des entiers de valeurs différentes – nous devrons donc être vigilant sur ce point…

Dans une classe que vous appellerez « Set », qui contiendra un attribut ArrayList d'Integer, implémentez les méthodes statiques suivantes :

1- ***public static Set singleton (Integer x)*** qui à partir d'un objet de type Integer x, retourne un objet de type Set ne contenant que cet élément x uniquement.

2- ***public Integer cardinal ()*** qui renvoie le nombre d'éléments contenu dans le Set

3- ***public Integer get (int i)*** qui permet d'obtenir l'entier se trouvant en index i.

4- ***public static boolean contains (Integer x)*** qui renvoie true si l'entier x est contenu dans l'ensemble s et false sinon.

5- ***public void add (Integer x)*** qui ajoute l'élément de type Integer x à la fin de du Set. Attention : bien vérifier que l'ArrayList ne contient x.

6- ***public void print ()*** qui permet d'afficher un Set complet.

Exemple : Si s contient les entiers 1, 2, 3, 4, 5 nous aurons la sortie sur la console sous la forme suivante : {1, 2, 3, 4, 5}.

7- ***public Set intersect*** (***Set s)*** qui retourne l'intersection de l'ensemble s1 avec l'ensemble s2, c'est-à-dire l'ensemble des éléments qui sont dans s1 et dans s2.

Exemple : l'intersection de {1, 2, 3, 4, 5} et de {1, 3, 5, 7 ,9} est l'ensemble {1, 3, 5}.

8- ***public Set union*** (***Set s)*** qui réalise l'union du set courant avec l'ensemble s, c'est-à-dire l'ensemble des éléments qui appartiennent à s1 ou à s2.

Exemple : l'union de {1, 2, 3} et de {2, 3, 4, 5} est l'ensemble {1, 2, 3, 4, 5}.

Annexe : rappel de la structure d'une classe

```
class Personne
{
        // variable d'instance donc privées
        private String nom;
        private int age;

        // variable de classe donc static
        static private int nb=0;

        // constructeur : de même nom que la classe
        // éventuellement plusieurs constructeurs
        public Personne(String lenom, int lage){
                nom=lenom;
                age=lage;
        nb++;
   }

        public Personne(){
                nom="Le Gac";
                age=20;
                nb++;
        }

        // Méthodes d'instance
        public int getAge(){
                return this.age;
        }

        public String getNom(){
                return this.nom;
        }

        public void setNom(String nouveauNom){
                this.nom=nouveauNom;
        }

        public void setAge(int nouvelAge){
                this.age = nouvelAge;
        }

        // Méthode de classe
        static public int obtenirNb(){
                return nb;
```

```
        }

        // éventuellement une méthode main pour tester la classe
        static public void main(String args[]){
                ...
        }

}
```