UNIVERSITE DU 8 MAI 1945 Faculté des Mathématiques et de l'Informatique Et des Sciences de la Matière
DEPARTEMENT D'INFORMATIQUE

D.Bouroualeh 2017 – 2eme LI

Examen Final POO -2H-

Exercice 1.

1. Ecrire la classe Siege avec les membres:
boolean occupe: indique si le siège est occupé, false à la création;
cl : un objet de type Client, initialisé à null a la création du siège;
numSiege: un entier indiquant le numéro du siège.
Libre(): renvoi true si le siège est libre, false sinon.
reserver(Client c): une méthode qui marque le siège comme occupé et lui affecte sont client.
Le constructeur de Siege a comme paramètre le numéro du siège.

2. écrire la classe Wagon: un Wagon dispose d'une collection de Sieges et un numéro de wagon. Cette classe dispose de deux constructeurs. On crée un objet Wagon en spécifiant son numéro et le nombre des sièges, c'est aussi à l'intérieur de ce constructeur que seront créés et ajoutés les sièges avec des numéros successifs 1,2,3...nombreSiege. Ont peut aussi créer un Wagon en spécifiant uniquement sont numéro, dans ce cas le Wagon aura obligatoirement 10 sièges (écrire ce deuxième constructeur en une seule instruction).
boolean complet() : renvoi true si tous les sièges sont réservés, false sinon.
boolean reserverPlace(Client c): réserve le premier siège libre dans le wagon et renvoi true (utilise la méthode reserver(...) de Siege); elle renvoi false si aucun siège n'est libre dans le wagon.

3. écrire la classe Train: un train dispose d'une collection de Wagons. Un constructeur pour initialiser la liste des Wagons (la liste des Wagons en paramètre).
Des méthodes:
.... chercherWagon(int numeroWagon): qui cherche dans la collection et renvoi un pointeur sur le wagon du train correspondant au paramètre numeroWagon. Renvoi null si elle ne trouve pas.
boolean reserver(Client c, int numeroWagon): réserve sur le premier siège disponible dans le wagon spécifié. Renvoi false si tous les sièges sont occupés.
Il est aussi possible de réserver pour un client en prenant le premier siège disponible dans n'importe quel wagon du train.
..... Joindre(.....): cette méthode renvoi un nouveau Train avec les wagons du train sur lequel est invoquée cette méthode et ceux d'un deuxième Train en paramètre.

4. écrire une classe de test avec un train de deux wagons et réservez pour un client.

Exercice 2. Un produit dispose d'un prix, d'un nom et d'une Marque. Une **méthode** prixTaxe renvoi le prix du produit plus (+) la taxe calculée par la marque du produit sur le prix du produit. La classe produit a un constructeur avec toutes les valeurs des attributs comme paramètres. Certain produits sont dit de luxe leur prixTaxe est calculé de la même façon qu'un produit mais est multiplier par 1.1 (10%). De plus, chaque produit de luxe a un taux de bénéfice. Pour ce type de produit on peut invoquer une méthode prixBenefice(....) qui renvoi le prixTaxe du produit de luxe augmenté par le bénéfice.
Q1. Ecrire les codes nécessaires, les constructeurs nécessaires en ayant tous les attributs privés et en ne rajoutant aucune autre méthode.
Q2. Ecrire une classe magasin qui dispose d'une collection de produits avec les méthodes:
sommePrixTax(): renvoi la somme des prixTaxe de tous les produits;
sommePrixTax(boolean avecBonif): renvoi la somme des prixTaxe de tous les produits, dans le cas d'un produit de luxe c'est le prix benefice qui sera ajouté à la somme.

```
class Client{
      public Client(int i) {
            codeClient=i;}

      int codeClient;
}
class Marque{
      double taxe;
      double taxer(double prix){return prix*taxe;}
      public Marque(double tax) {
            taxe = tax;
      }

}
       ArrayList<Desi> elements; // déclaration du référent
elements = new ArrayList<Desi>(); // création d'objet ArrayList typé

       elements.add(……); //ajout element
int s=elements.size();// récupère le nombre d'éléments dans la liste

            …. value = elements.get(3); // récupère l'élément d'indice 3
for(Tarte t:Plateau) {.. t.det();…..}
```