TP Algo/prog

Sujet1

On considère un ensemble de villes ainsi que les distances en kilomètres entre ces villes. L'objectif de cet exercice est de modéliser d'une part les voyages possibles entre ces villes (par exemple en train) ainsi que leurs prix, et d'autre part les voyageurs effectuant ces voyages.

1) Définir la classe *Voyage* permettant de modéliser un voyage (trajet) caractérisé par la ville de *départ* la ville d'*arrivée*, la *distance* entre les deux villes, et la *date* du départ. Cette classe doit contenir :
   - *a)* Un constructeur prenant en argument 4 paramètres permettant d'initialiser les données de la classe
   - *b)* Une méthode d'affichage des caractéristiques du voyage.
   - *c)* Redéfinir la méthode *toString* définie dans la classe *Object*
   - *d)* Une fonction *prix* permettant de calculer le prix du voyage selon les règles :
     - Jusqu'à 100 km : le prix du kilomètre est 0.15 €
     - Entre 100 et 200 km : le prix du kilomètre est 0.10 €
     - Au-delà de 200 km : le prix du kilomètre est 0.07 €

Par exemple pour un trajet de 360 Km : 100*0.15+100*0.10+160*0.07 €

2) Définir la classe *Voyageur* qui permet de modéliser un voyageur caractérisé par son trajet *aller*, son trajet *retour* (le retour n'est pas forcément à la ville de départ), et le *prix* du billet dont il est en possession.
   - *a)* Un constructeur permettant d'initialiser les différentes données
   - *b)* Une méthode d'affichage des caractéristiques du voyageur
   - *c)* Une fonction *enRegle* qui indique si le voyageur est en règle : cette fonction retourne *faux* si le prix du billet en possession du voyageur est inférieur au prix qu'il aurait dû payer, et *vrai* sinon. Dans le cas où le voyageur est en infraction, un message sera affiché indiquant l'amende que le voyageur doit payer (5 fois la différence entre le prix de son billet et prix du voyage). Dans le cas où le prix de son billet est trop élevé, un message indiquera le trop perçu.

2) Définir une classe *VoyageurAvecReduc* permettant de gérer les voyageurs qui peuvent bénéficier d'une réduction compte tenu de leur *age*. Le calcul du prix se fait de la façon suivante :
   - Si le voyageur a moins de 25 ans, il a droit à 25% de réduction
   - Si le voyageur a plus de 60 ans, il a le droit à 40% de réduction
   - Sinon, il paye le plein tarif

   Cette classe contiendra :
   - Un constructeur
   - Une méthode *prix* permettant de calculer le prix selon les règles ci-dessous
   - Redéfinir la méthode *enRegle* pour qu'elle prenne en compte les nouvelles règles

3) Définir une classe *TestVoyage* permettant de contrôler un voyageur de 20 ans qui fait le trajet (aller/retour) entre Lens et Paris le 11/09/2019 sachant que la distance est de 200 Km.

Sujet 2

Une usine fabrique des pièces. On distingue deux types de pièces : pièces simples et pièces complexes. Les pièces complexes sont composées de plusieurs pièces (déjà fabriquées). Pour simplifier, on suppose qu'une pièce complexe n'est pas composée de plus d'une pièce d'un autre type ; par exemple, la pièce de code p1 pourra être composée d'une pièce de code p2 et d'une pièce de code p3 (mais pas de 5 pièces de code p3). En plus, une pièce complexe peut être composée de 10 pièces au plus. On s'intéresse au temps nécessaire (en minutes) pour la fabrication de ces pièces. Pour les pièces complexes, on connait directement le temps nécessaire à l'assemblage des pièces (déjà fabriquées) mais on voudrait savoir le temps total nécessaire, y compris le temps qu'il a fallu pour fabriquer les pièces assemblées. Chaque pièce simple est caractérisée par son *nom* et le *temps* nécessaire à sa fabrication. Une pièce complexe est caractérisée, en plus, par un *tableau* décrivant les pièces qui la composent et le *nombre* de ces pièces. Les classes doivent être écrites en Java.

1. Écrire la classe **Piece** qui contient un ***constructeur*** et deux fonctions : la première ***getTemps*** retourne le temps de fabrication et la deuxième ***getTempsTotal*** retourne le temps total de fabrication (pour une pièce simple ces deux fonctions sont identiques).
2. Écrire la classe **PieceComplexe** qui est une sous-classe de la classe *Pièce*. Elle contient un ***constructeur*** qui initialise les données. Elle contient aussi la fonction ***getTempsTotal*** qui calcule et retourne le temps total nécessaire à la fabrication de cette pièce. Cette classe doit contenir également une procédure ***ajout*** qui accepte un paramètre de type Piece et qui ajoute la pièce passée en paramètre au tableau des pièces.
3. Ecrire la classe TestPiece qui permet de tester ces classes. On doit pouvoir créer des pièces simples et des pièces complexes. Voici les objets que doit créer la méthode main() (le temps de fabrication de la pièce elle-même est donné entre parenthèses) : 2 pièces simples p1 (15) et p2 (20), 2 pièces complexes p3 (10) et p4 (25). p3 est composé des pièces p1 et p2 ; p4 est composé des pièces p1, p2 et p3.