

SERIE DE TP N° 4

Matière : Programmation Java

Section : L2 Informatique

Exercice 1 :

Ecrire un programme qui simule le mouvement d'un robot simple. Un tel robot occupe une certaine position (X, Y), il a une orientation parmi (Nord, Est, Sud, Ouest), il est initialisé à une position et une orientation donnée, il peut tourner à droite ou bien à gauche et il peut avancer d'un pas.

La classe **Robot** doit avoir les attributs :

- x, y de type int
- orientation qui peut prendre une valeur constante entre 1 et 4 : 1 pour Nord ; 2 Est pour, 3 pour Est et 4 pour Ouest.

Ajouter les constructeurs que vous jugez nécessaires.

Ajouter les méthodes suivantes :

- **tournerADroite**, **tournerAGauche** qui modifient l'orientation du robot sans changer sa position.
- **avancer(int d)** qui modifie la position du robot selon son orientation actuelle.
- **afficher** qui affiche la position et l'orientation du robot.
- d'accès aux attributs (get et set).

Ecrire aussi un programme qui teste la classe **Robot**.

Exercice 2 :

1. Ecrire la classe **Rectangle** qui possède trois attributs :
 - sommet (de type **Point** (voir TP3) et qui représente le point en haut et à gauche)
 - deux réels qui représentent la longueur et la largeur du rectangle.
2. Ajouter les constructeurs que vous jugez nécessaires.
3. Ajouter les méthodes d'accès aux attributs.
4. Ajouter les méthodes **surface** et **perimetre** qui permettent respectivement de retourner la surface et le périmètre d'un rectangle.

-
5. Ajouter la méthode **afficher** qui permet d'afficher les informations d'un rectangle.
 6. Ajouter la méthode **appartient(Point p)** qui retourne un booléen indiquant si le point p passé comme argument se trouve dans le rectangle ou non.
 7. Créer un programme principal dans lequel vous testez la classe Rectangle et ses différentes méthodes.

Exercice 3 :

1. Créer une classe **Livre** avec les attributs suivants :
 - titre (String)
 - auteur (String)
 - nbPages (int)
 - nbExemplairesDisponibles (int)
2. Ajouter les constructeurs que vous jugez nécessaires.
3. Ajouter les méthodes suivantes :
 - emprunter() décrémente *nbExemplairesDisponibles*, si le livre n'est pas disponible elle affiche un message d'erreur.
 - retourner() incrémente *nbExemplairesDisponibles*
 - afficher() qui affiche les informations d'un livre.
4. Créer une classe **Bibliotheque** contenant :
 - un tableau de type **Livre** dont la taille est initialisée (par exemple à 10) au moment de la création.
 - nbLivres (int) représentant le nombre de livres se trouvant à la bibliothèque
5. Ajouter les constructeurs que vous jugez nécessaires.
6. Ajouter les méthodes suivantes :
 - ajouterUnLivre(Livre l) permettant d'ajouter le livre l à la bibliothèque si le tableau n'est pas plein.
 - afficherTous() qui affiche tous les livres.
 - chercherParTitre(String t) qui retourne le livre correspondant ou **null** s'il n'existe pas.
7. Ecrire un programme principal dans lequel vous devez créer une bibliothèque, ajouter des livres, emprunter quelques livres, les retourner, afficher leurs informations.

Exercice 4 :

1. Créer une classe **Produit** avec les attributs suivants :
 - nom (String)
 - prixUnitaire (double)
 - quantite (int)
2. Ajouter les constructeurs que vous jugez nécessaires.
3. Ajouter les méthodes suivantes :
 - ajouterQuantite(int qte) : augmente la quantité du produit en stock.
 - retirerQuantite(int qte) : diminue la quantité si elle est suffisante, sinon affiche un message d'erreur.
 - valeurStock() : retourne la valeur totale du stock (prixUnitaire \times quantite).
 - afficher() : affiche les informations du produit (nom, prix unitaire, quantité et sa valeur totale de stock).
 - Getters et setters.
4. Créer une classe **LigneDeFacture** contenant :
 - Le produit ajouté dans une ligne de facture.
 - La quantité vendue
5. Ajouter les constructeurs que vous jugez nécessaires.
6. Ajouter les méthodes suivantes :
 - sousTotal() qui calcule et retourne le sous-total d'une ligne de facture (prix unitaire du produit \times quantité vendue)
 - afficher() : affiche les informations de la ligne de facture (nom du produit, son prix unitaire, la quantité vendue et le sous-total).
7. N'oublier pas de mettre à jour la quantité d'un produit s'il est ajouté dans une ligne de facture.
8. Créer une classe **Facture** contenant :
 - codeFacture (String).
 - un tableau de type **LigneDeFacture** dont la taille est initialisée (par exemple à 10) au moment de la création.
 - nbLigneDeFactures (int) représentant le nombre de lignes de facture ajoutées à la facture.
9. Ajouter les constructeurs que vous jugez nécessaires.
10. Ajouter les méthodes suivantes :

- ajouterLigneDeFacture (LigneDeFacture ldf) : ajoute une ligne de facture à la facture si le tableau n'est pas plein.
- total() : retourne le montant total de la facture (somme des sous-totaux de toutes les lignes de facture qui sont ajoutées à la facture).
- afficher() : affiche le code de la facture, la liste des produits se trouvant dans les lignes de facture et le total général.

11. Écrire un programme principal dans lequel vous devez :

- Créer plusieurs produits
- Créer des lignes de facture et y ajouter les produits (chaque produit dans une ligne de facture)
- Afficher la facture complète et son total.

FACTURE			
CODE FACTURE: F001			
PRODUIT	QUANTITÉ	PRIX UNITAIRE	SOUS-TOTAL
Stylo	10	1,50	15,00
Cahier	5	3,20	16,00
Classeur	2	5,00	10,00
TOTAL			41,00




Figure 1. Exemple d'une facture avec des lignes de facture et des produits

Exercice 5 :

L'objectif de cet exercice est de comprendre le rôle et la portée des modificateurs de visibilité (*public*, *protected*, *private*, et *sans modificateur*) selon le contexte d'accès :

- dans la même classe,
- dans une sous-classe du même package,
- dans une classe non héritée du même package,
- dans une sous-classe d'un autre package,
- dans une classe non héritée d'un autre package.

Créer les classes se trouvant dans le diagramme de classes présenté la figure 2.

Dans la classe A déclarer les attributs x, y, w, et z respectivement avec les modificateurs de visibilité *private* (-), *public* (+), *protected* (#), et *sans modificateur*, les initialiser avec des valeurs simples telles que 1, 2, 3 et 4 dans un constructeur sans paramètres.

Dans les cinq classes définir une méthode appelée *testAcces()* dans lesquelles on essaye d'accéder aux différents attributs de la classe A pour afficher leurs valeurs soit directement (si possible) soit en instanciant la classe A.

L'héritage en java se fait par le mot clé *extends*.

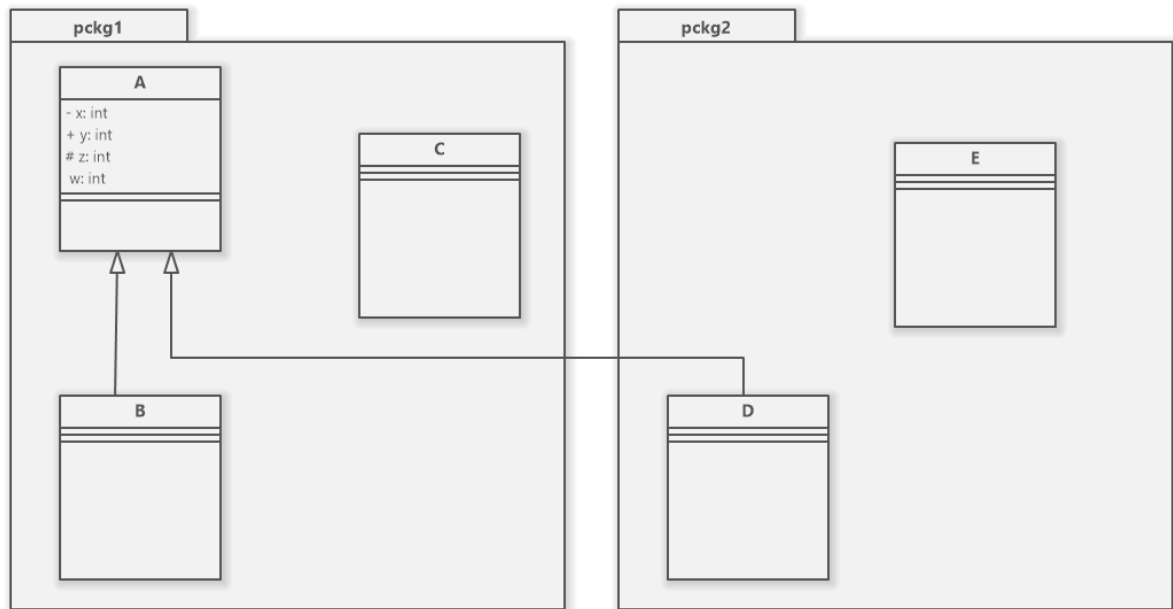


Figure 2. Diagramme de classes [Modificateurs de visibilité]