

TD N°2 POO : les classes Java

Installer l'environnement JAVA suivant :

- L'Editeur NetBeans
- Le JDK

Exercice N° 01 :

1. Écrire un programme affichant "Hello World" sachant que l'instruction permettant d'afficher un chaîne à l'écran est : **System.out.println("[chaîne]");**
2. Écrire un programme affichant 10 fois "Hello World"
3. Écrire un programme affichant le premier argument donné en lançant la classe.
4. Ecrire un programme qui lit deux entiers passés en paramètres sur la ligne de commande et affiche leur somme, leur différence, leur produit et leur quotient. Attention, on ne peut pas diviser par 0.
5. Ecrire un programme qui lit une note passée en paramètre et qui indique si elle est suffisante pour réussir un examen. On pourra vérifier l'existence de la note et arrêter le programme si ce n'est pas le cas en utilisant l'instruction **System.exit(1)**.

Exercice N° 02 :

1. Écrire une classe Voiture possédant les propriétés **Age** et **Kilometrage** ainsi que la méthode **getKilometrage()** qui renvoie le kilométrage courant du véhicule. La classe Voiture aura deux constructeurs : le premier prenant uniquement l'âge et l'autre l'âge et le kilométrage. Ces deux constructeurs n'auront pas de code redondant. Par défaut, le kilométrage sera de zéro km.
2. Écrire une classe Test ayant une méthode main et instanciant deux voitures. Affichez l'âge et le kilométrage de ces voitures.
3. Définir dans la classe Voiture un attribut primitif NbRoues statique qui donne le nombre de roues.
4. Dans Test, fixer la valeur de NbRoues d'une voiture à 4 et afficher la valeur NbRoues des autres instances de Voiture.

Exercice 3 :

Ecrire une classe *Médicament* contenant :

- 1) Trois attributs privés :
 - ✚ **Nom** : nom du médicament.
 - ✚ **Prix** : le prix du médicament.
 - ✚ **Stock** : la quantité stockée du médicament.
- 2) Les constructeurs :
 - ✚ Un constructeur permettant de créer un médicament à partir du son nom.
 - ✚ Un constructeur permettant de créer un médicament à partir du son nom et du prix.
 - ✚ Un constructeur permettant de créer un médicament à partir du son nom et du stock.
 - ✚ Un constructeur permettant de créer un médicament à partir du son nom, son prix et son stock.
- 3) Une méthode *augmenterStock()* recevant en paramètre la quantité en stock et permettant d'augmenter le stock.
- 4) Une méthode *diminuerStock()* recevant en paramètre la quantité en stock et permettant de diminuer le stock.
- 5) Les méthodes : *getStock()*, *getPrix()*, *getNom()* permettant de récupérer le nom, le prix et le stock.
- 6) Une méthode *affiche()* permettant d'afficher les information d'un médicament sous la forme suivante : Le médicament *#Nom#* a un prix de : *#prix#* et un stock de : *#stock#*.
- 7) Définir des méthodes permettant de modifier les attributs de la classe.
- 8) Définir une méthode main permettant de :
 - ✚ Créer les médicaments suivants:
 - ✚ Fervex : stock : 20, prix : 5DT
 - ✚ Aspirine : 1500 DT
 - ✚ Adol.
 - ✚ Ajouter la quantité 40 pour aspirine. Afficher le résultat de l'ajout.
 - ✚ Que devient le stock du Fervex si on l'ajoute la quantité 30. Afficher le résultat obtenu.
 - ✚ Modifier le prix du Aspirine à 1200. Afficher le prix modifié.

Exercice 4 :

Définir une classe nommée `MonCompte` destinée à gérer le ou les comptes d'une personne.

Cette classe possède les deux attributs suivants :

- # `nom` : nom de la personne
- # `solde` : attribut privé de type double.

et les méthodes suivantes :

- # `MonCompte()` : Crée un compte vide.
- # `MonCompte(double solde)` : Crée un compte avec un solde initial.
- # `MonCompte(String nom, double solde)` : Crée un compte avec un solde initial et un nom de personne.
- # les méthodes getters et setters nécessaires
- # `void retirer(double montant)` : Retire une somme d'argent du solde (le montant doit être inférieur ou égal au solde)
- # `void deposer(double montant)` : Dépose une somme d'argent sur un compte.
- # `void transferer(double montant, MonCompte compte)` : Transfère une somme d'argent du compte courant vers un autre compte.
- # `public static void main(String[] args)` dans laquelle on :
 - 1) Créer un premier compte `b1` en spécifiant le solde initial (égal à 100)
 - 2) Créer un second compte `b2`
 - 3) Afficher les soldes de ces comptes
 - 4) Effectuer les transactions suivantes :
 - Déposer 30 sur `b2`
 - Retirer 45 de `b1`
 - Déposer 20 sur `b1`
 - Déposer 100 sur `b2`
 - Retirer 10 de `b2`
 - 5) Transférer 30 du compte `b2` vers `b1`.
 - 7) Affiche les soldes finaux pour ces comptes.