```
public int getId(){
    return id;
}

public String getNom(){
    return nom;
}

public int getQt(){
    return qt;
}

import java.util.List;
Class GestionStock{
    private List <Produit> lp = new ArrayList <Produit>();

    public void ajouterProduit(Produit P){
        lp.add(P);
    }

    Public void AfficherDetailProduit(){
        for(int i = 0; i < lp.size; i++){
            System.out.println(lp.get(i).toString();
        }
    }

    Public void trouverProduit(int idP){
        if(Produit P: lp){
            if(P.getId() == idP){
                return P;
            else
                return Null;
            }
        }
    }
}
```

lp.get(i) ←

return Null;

for(int i=0; i<lp.

if(lp.get(i).

EX:

Définir class vecteur Avec des Attributs int (x,y,z) privés

- Définir un ou plusieurs constructeurs
- Méthode (~~Prodscal~~) somme de deux vecteurs
- Méthode Prodscal de 2 vecteurs
- Méthode Affiche
- Méthode main pour le Test.

$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

Reponse:

```
Public class Vecteur{
    Private int x,y,z;
    Public Vecteur(){ }
    Public Vecteur (int x, int y, int z){
        this.x = x ;
        this.y = y ;
        this.z = z;
    }
    Vecteur Somme (Vecteur V){
        Vecteur W = new Vecteur();
        W.x = V.x + this.x;
    }
    Public Vecteur somme( Vecteur V1, Vecteur V2){
        Vecteur V3=new Vecteur( );
        V3.x  int x = V1.x + V2.x ;
        V3.y  int y = V1.y + V2.y ;
        V3.z  int z = V1.z + V2.z ;
        Vecteur V3 = new vecteur(x,y,z);
        return V3;
    }
```

```
public int ProdScal (Vecteur V1, Vecteur V2){
    { int r;
      r = V1.x * V2.x + V1.y * V2.y + V1.z * V2.z ;
      return r;
    }
Public void Affiche( )
      { System.out.println (x+" "+y+" "+z
      }
class Test {
    main( ){
    { Vecteur V1 = new Vecteur (1,2,3);
      Vecteur V2 = new Vecteur (4,5,6);
Vecteur W = V1. Somme (V1, V2);
    Vecteur W=V1. Somme (V2);
        V1. affiche ;
```

TD3 – POO Java

Exercice 1 : Trouvez et corrigez les dix (10) erreurs dans les classes suivantes :

```
1.  package p1;
2.  class A {
3.  private int a;
4.  public A(int a){
5.  this.a=a;
6.  }
7.  public void m(){
8.     System.out.println(a);  }
9.  }
```

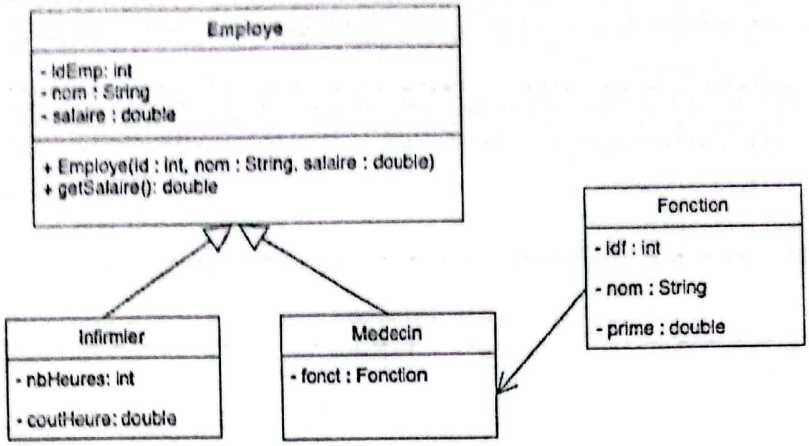
```
10. package p2;
11. public class B extends A {
12. private int b;
13. public B (int a, int b){
14.    this.b = b;
15. }
16. void m1(){
17.     System.out.println(a + b); }
18. }
```

```
19. package p2;
20. public class C extends B {
21. public void m(){ System.out.println(a);}
22. public void m(){ System.out.println(b);}
23. }
```

```
24. package p3;
25. public class Test {
26. public static void main(String args[]){
27. A a=new A(5);
28. B b=new B(10,20);
29. b=a;
30. a.m1();
31. Object d1 = (B) a;
32. a=b;
33. a.m();
34. Object d2 = (B) a; 9. ((B) d2).m(); }
```

Exercice 2 :

Dans cet exercice nous allons modéliser le système d'information représenté par les classes suivantes :



NB. Respecter les principes de **l'encapsulation** de la manière la plus stricte.

1) Écrire les instructions Java qui permettent de définir la classe **Employe** avec les contraintes suivantes :
   - Le constructeur permet de fixer les valeurs respectivement, `idEmp`, `nom` et `salaire`
   - Prévoir un moyen d'obtenir la valeur de l'attribut salaire.
   - Écrire les instructions qu'il faut ajouter à la classe **Employe** pour que :

     `System.out.println(e)` affiche l'état détaillé du **`Employe e`**.

2) Écrire les instructions Java qui permettent de définir la classe **Infirmier** avec les contraintes suivantes :
   - Un infirmier est caractérisé par: `nbHeures` (entier) et le `coutHeure` (double)
   - Redéfinir la méthode **`+getSalaire():double`** qui renvoie le résultat de la multiplication de nbHeures par le coutHeure.

3) Écrire les instructions Java qui permettent de définir la classe **Medecin** avec les contraintes suivantes :
   - Un **Medecin** est caractérisé par sa fonction
   - Redéfinir la méthode **`+getSalaire():double`** qui doit renvoyer le résultat de

la somme de : le `salaire` et le `prime` donné par la classe **`Fonction`**.

4) Écrire les instructions Java qui permettent de définir la classe **`Fonction`** avec les contraintes suivantes :

   - Une fonction est caractérisée par : `idf`(entier), `nom` (chaine de caractère) et `prime` (double)

5) Écrire le programme principal qui permet de :

   - Créer deux objets, de la classe **`Infirmier`**, nommés : `inf1 et inf2`
   - Créer la liste « `listInf` » des infirmiers contenant les deux objets `inf1 et inf2`
   - Afficher l'état détaillé de tous les objets de la liste « `listInf` »
   - Créer deux objets de la classe **`Medecin`**, nommés `med1 et med2`, avec ses fonctions respectivement « `Assistant` » et « `chef service` ».
   - Créer la liste « `listMed`» des médecins contenant les deux objets `med1 et med2`.
   - Afficher l'état détaillé de tous les objets de la liste « `listMed`»