Formation Session 4 2018

Formateur: AMEUR .K
E-Mail: ameur.khadidja@univ-ouargla.dz

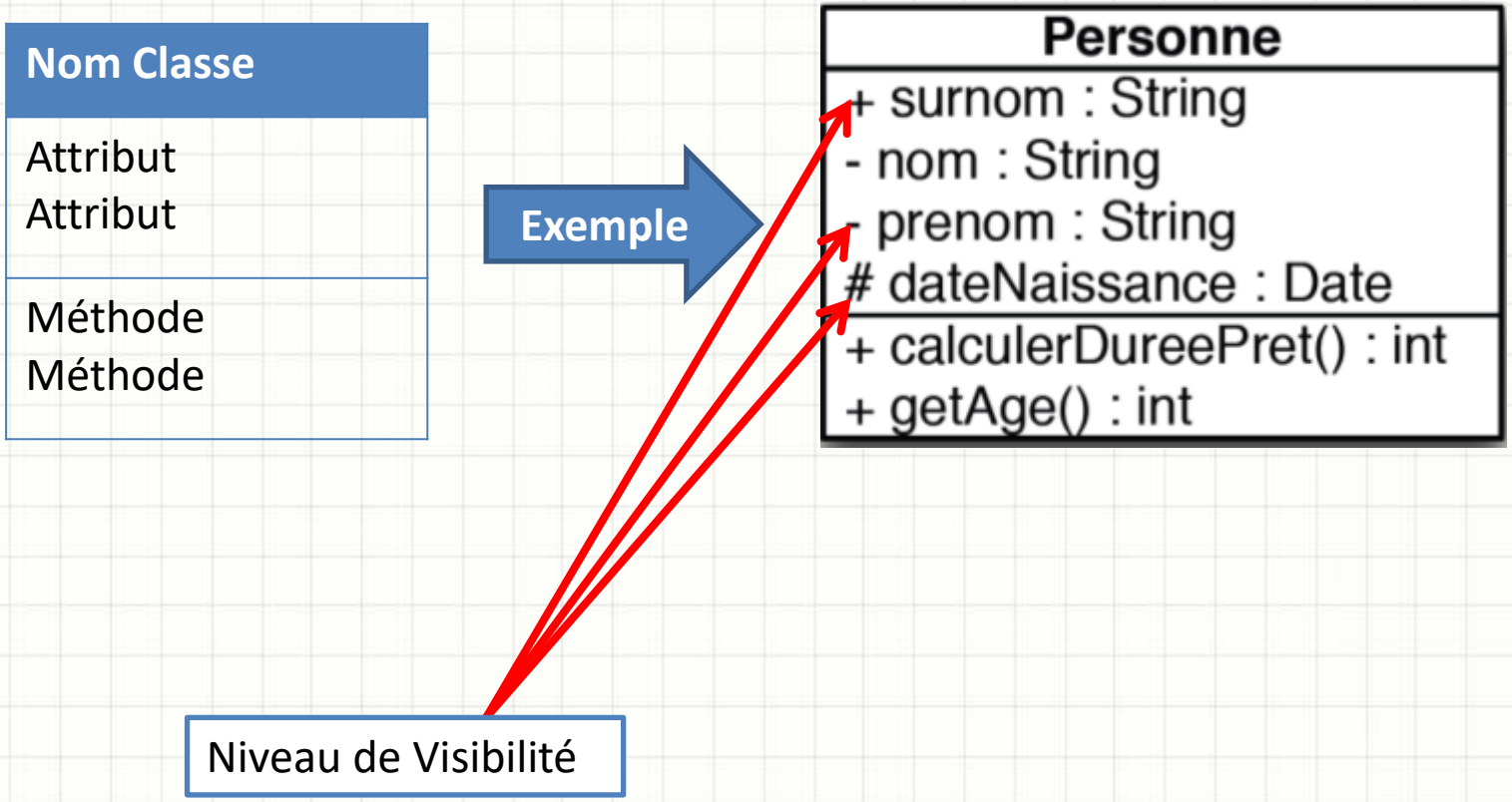
POO

Ameur .K

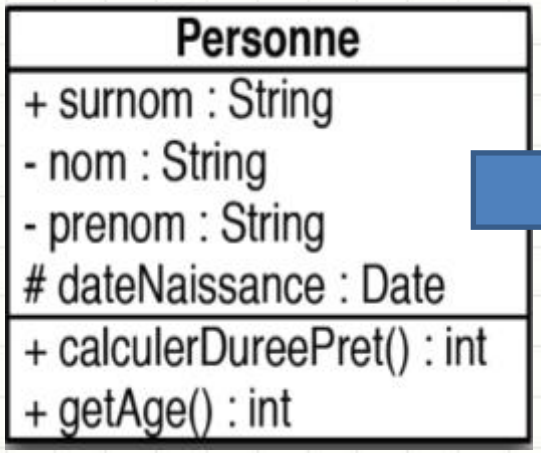
ameur.khadidja@univ-ouargla.dz

DIAGRAMME DE CLASSES UML/JAVA

UML Classe

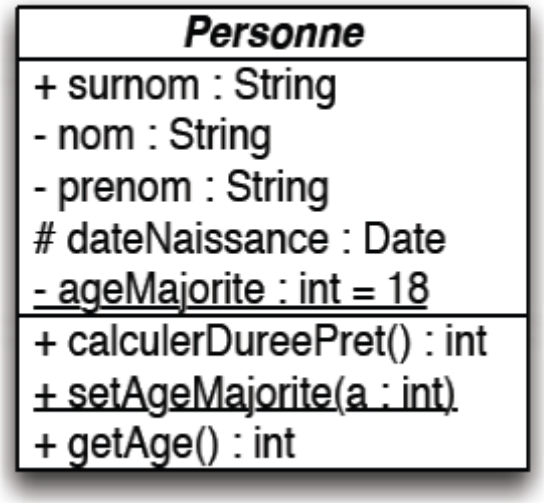

Nom Classe
Attribut
Attribut
Méthode
Méthode
Exemple
Personne
+ surnom : String
- nom : String
- prenom : String
dateNaissance : Date
+ calculerDureePret() : int
+ getAge() : int
Niveau de Visibilité

Java Classe




```java
public class Personne {
  public String surnom;
  private String prenom;
  private String nom;
  protected Date dateNaissance;
  public int calculerDureePret() {...}
  public int getAge() {...}
}
```

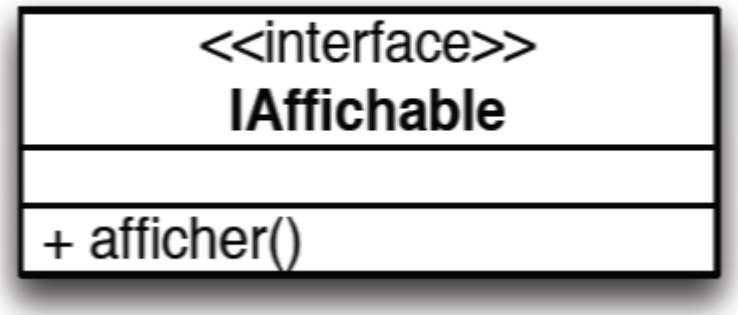
Classe Abstraite




```java
public abstract class Personne {
  public String surnom;
  private String prenom;
  private String nom;
  protected Date dateNaissance;
  private static int ageMajorite = 18;
  public int calculerDureePret() {...}
  public static void setAgeMajorite(int a) {...}
  public int getAge() {...}
}
```

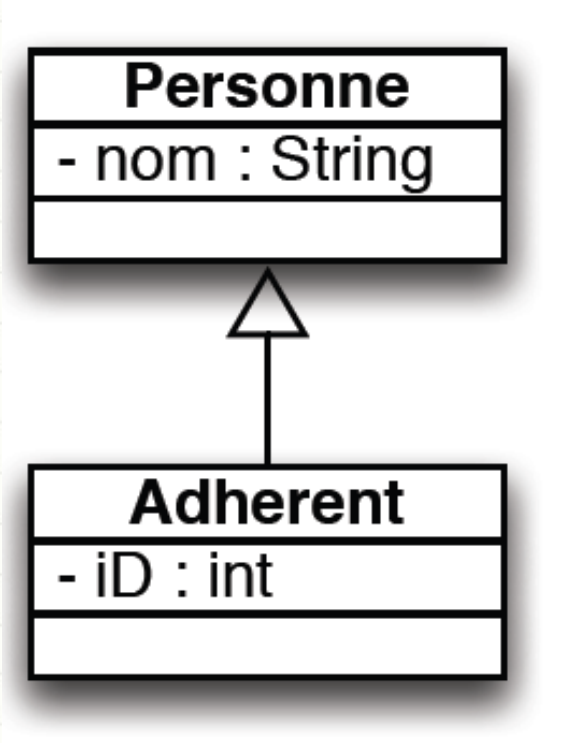
Les interfaces

```
interface IAffichable {
  void afficher();
}
```



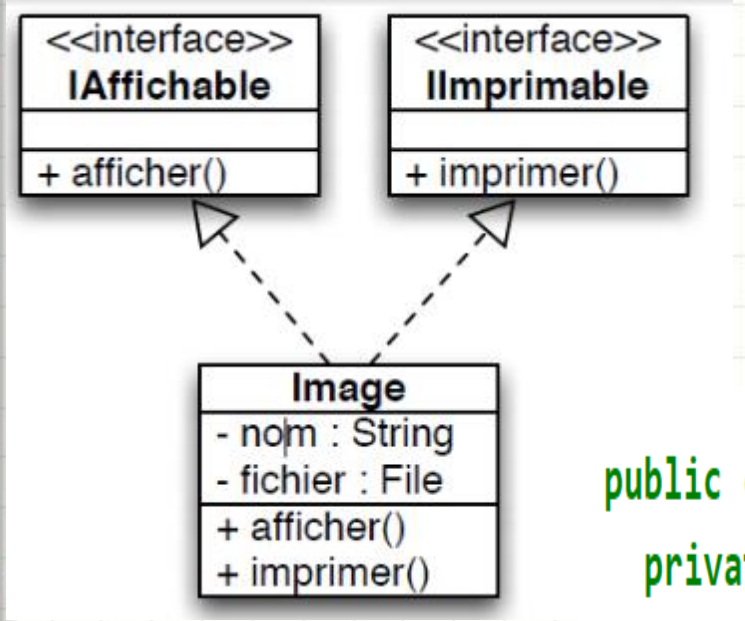
LES RELATIONS

Les relations: L'héritage




```java
public class Adherent extends Personne {
  private int iD;
}
```

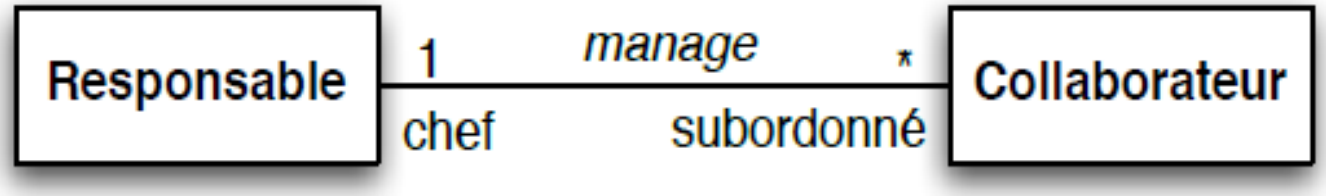
Les relations: La réalisation




```
public class Image implements IAffichable, IImprimable {
  private String nom;
  private File fichier;
  public void afficher(){...}
  public void imprimer(){...}
}
```

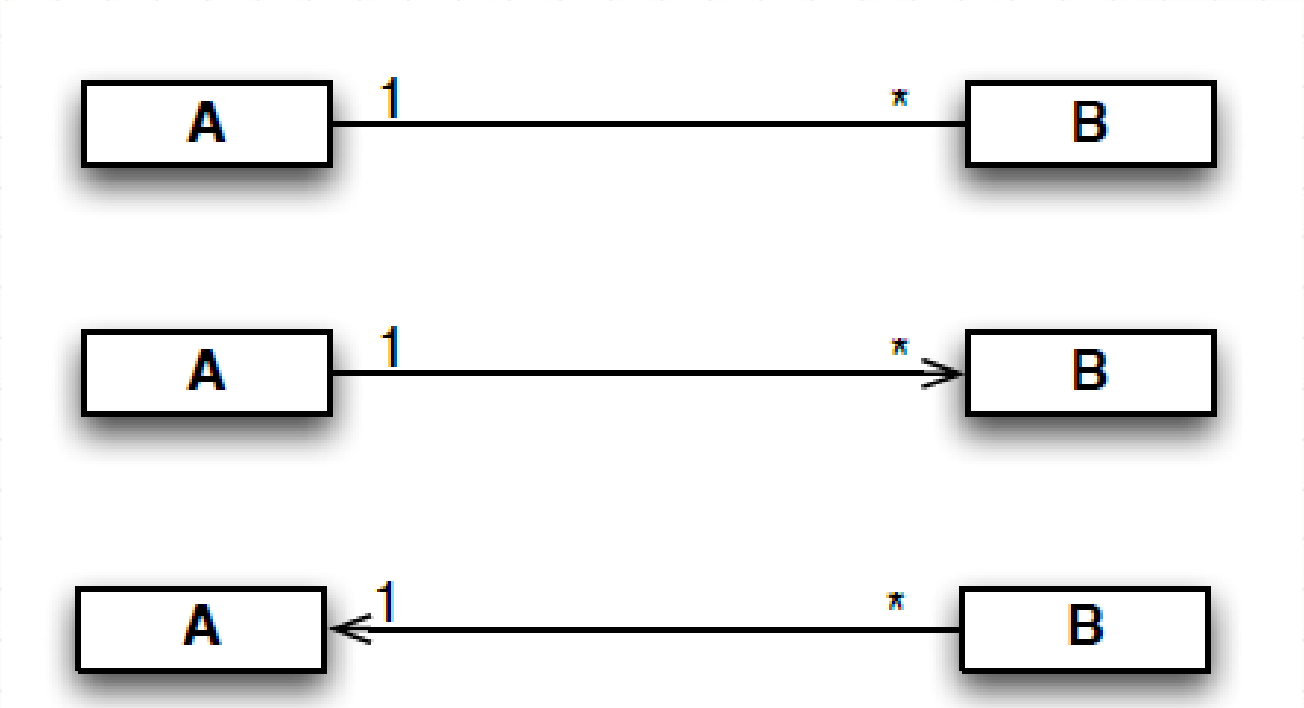
Les relations: Les associations

Multiplicité	Interprétation
1	un et un seul
0..1	zéro ou un
N	exactement N
M..N	de M à N
*	zéro ou plus
0..*	zéro ou plus
1..*	un ou plus



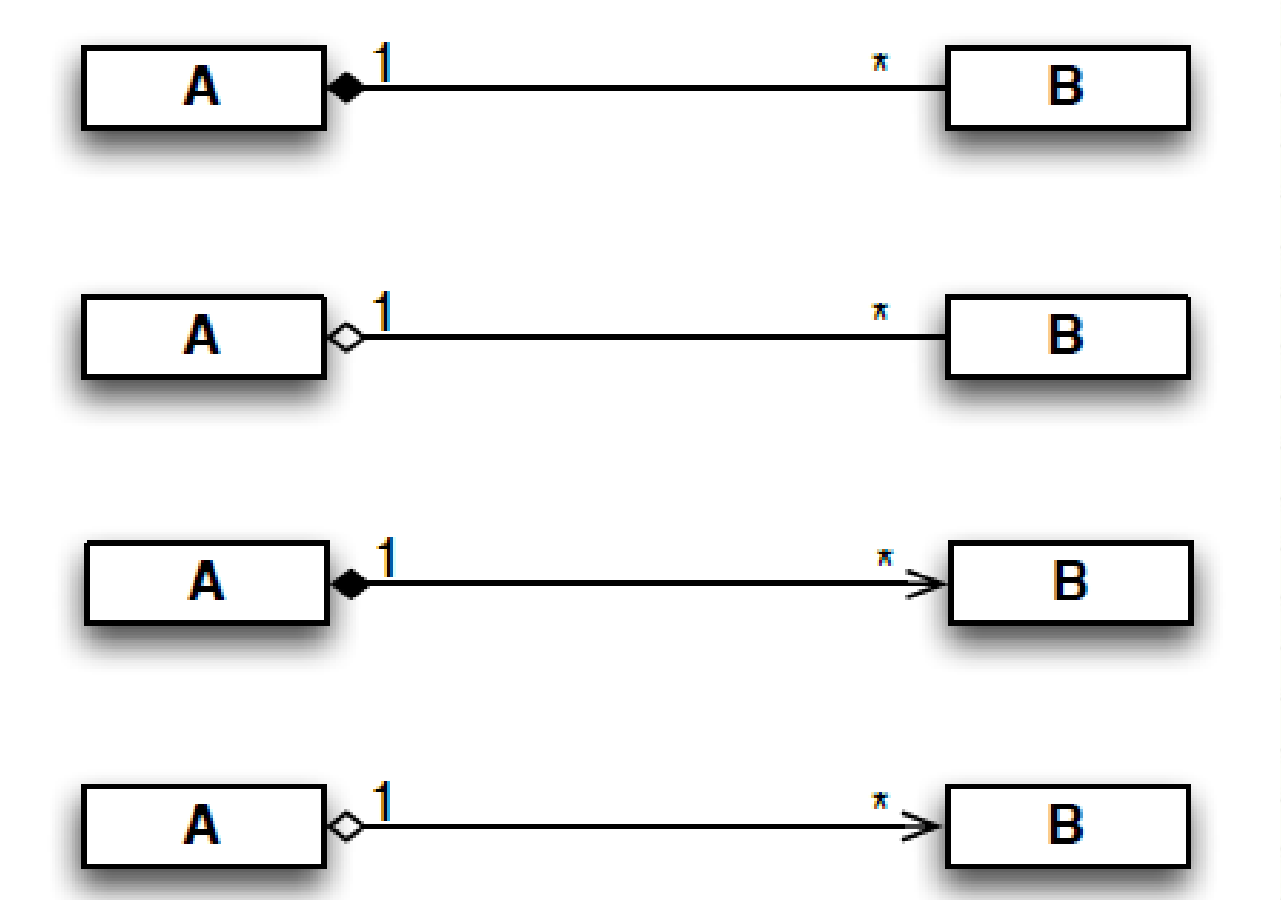
Les relations: Les associations

- **Direction des associations**

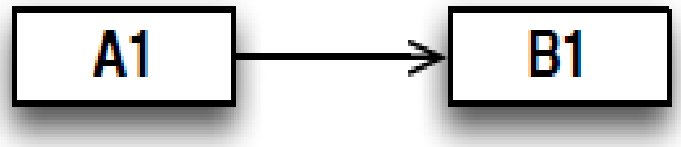


Les relations: Les associations

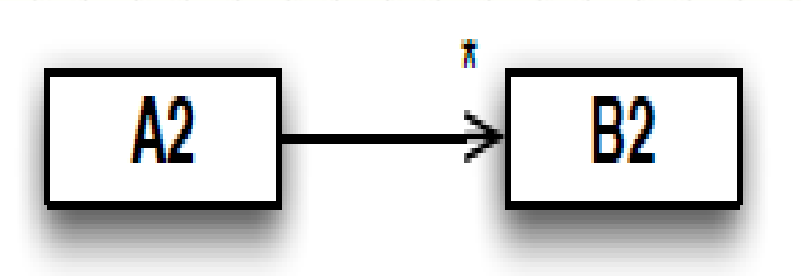
- **Agrégation et composition**



Les relations: Les associations 1

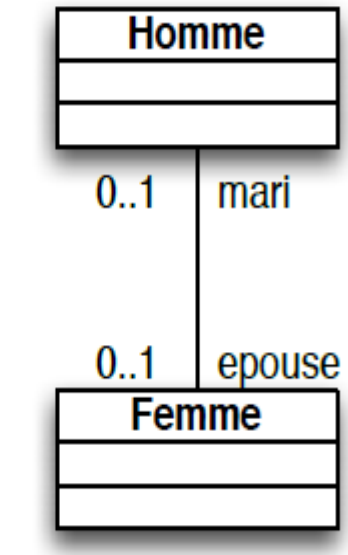


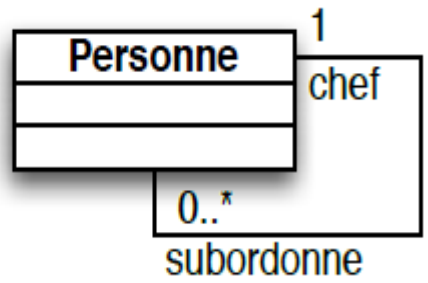

```
public class A1 {
  private B1 b1;
  ...
}
```




```
public class A2 {
  private ArrayList<B2> b2s;
  ...
}
```

Les relations: Les associations 2



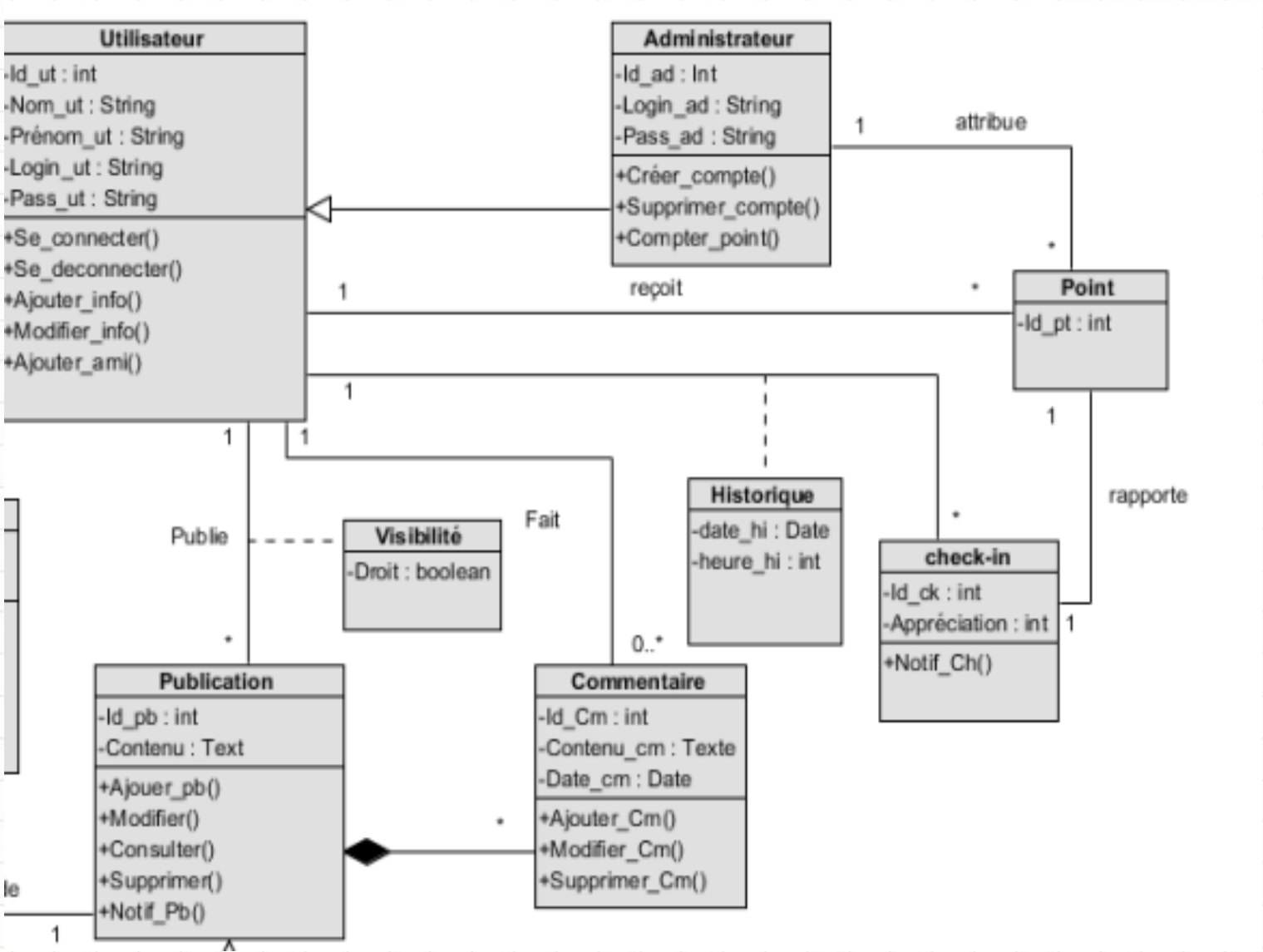


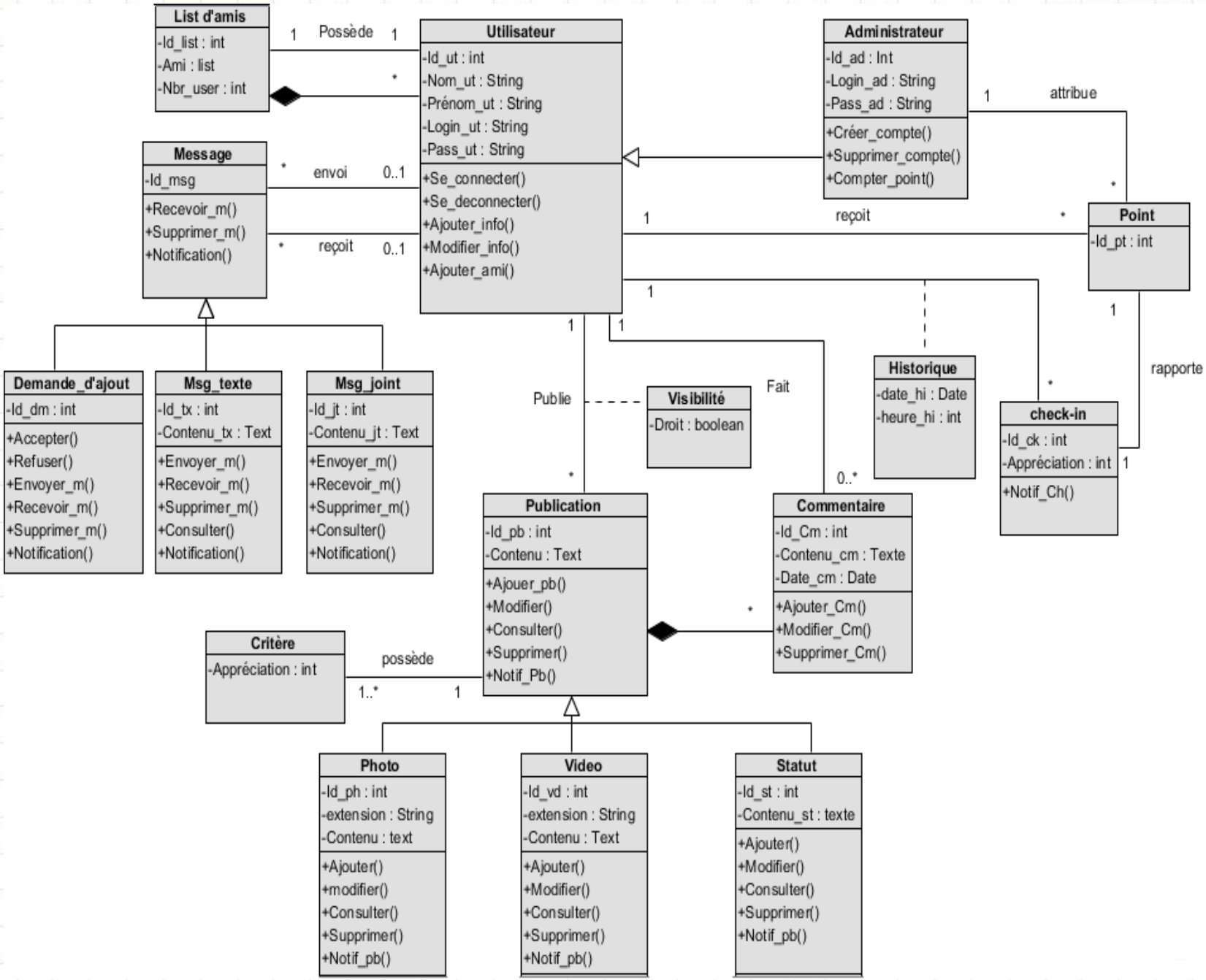

```java
public class Homme {
  private Femme epouse;
  ...
}
```

```java
public class Femme {
  private Homme mari;
  ...
}
```

```java
public class Personne {
  private ArrayList<Personne> subordonnes;
  private Personne chef;
  ...
}
```

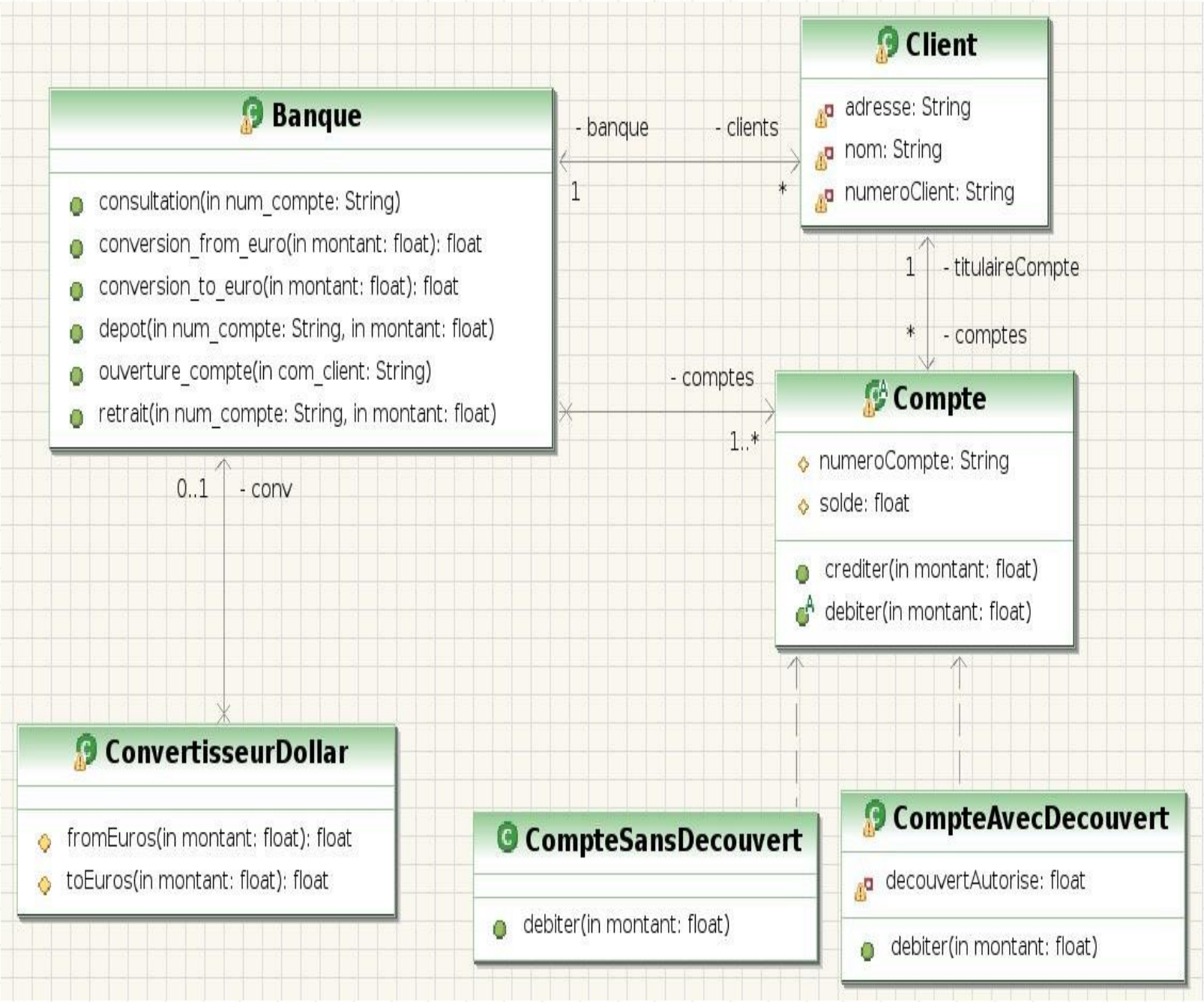
Exercice 1 :


Utilisateur
-Id_ut : int
-Nom_ut : String
-Prénom_ut : String
-Login_ut : String
-Pass_ut : String
+Se_connecter()
+Se_deconnecter()
+Ajouter_info()
+Modifier_info()
+Ajouter_ami()
Administrateur
-Id_ad : Int
-Login_ad : String
-Pass_ad : String
+Créer_compte()
+Supprimer_compte()
+Compter_point()
attribue
reçoit
Point
-Id_pt : int
Publie
Visibilité
-Droit : boolean
Fait
Historique
-date_hi : Date
-heure_hi : int
rapporte
check-in
-Id_ck : int
-Appréciation : int
+Notif_Ch()
Publication
-Id_pb : int
-Contenu : Text
+Ajouer_pb()
+Modifier()
+Consulter()
+Supprimer()
+Notif_Pb()
Commentaire
-Id_Cm : int
-Contenu_cm : Texte
-Date_cm : Date
+Ajouter_Cm()
+Modifier_Cm()
+Supprimer_Cm()


List d'amis
-Id_list : int
-Ami : list
-Nbr_user : int
1
Possède
1
*
Utilisateur
-Id_ut : int
-Nom_ut : String
-Prénom_ut : String
-Login_ut : String
-Pass_ut : String
+Se_connecter()
+Se_deconnecter()
+Ajouter_info()
+Modifier_info()
+Ajouter_ami()
Administrateur
-Id_ad : Int
-Login_ad : String
-Pass_ad : String
+Créer_compte()
+Supprimer_compte()
+Compter_point()
1
attribue
*
Message
-Id_msg
+Recevoir_m()
+Supprimer_m()
+Notification()
*
envoi
0..1
*
reçoit
0..1
1
reçoit
*
Point
-Id_pt : int
1
1
1
1
1
rapporte
Demande_d'ajout
-Id_dm : int
+Accepter()
+Refuser()
+Envoyer_m()
+Recevoir_m()
+Supprimer_m()
+Notification()
Msg_texte
-Id_tx : int
-Contenu_tx : Text
+Envoyer_m()
+Recevoir_m()
+Supprimer_m()
+Consulter()
+Notification()
Msg_joint
-Id_jt : int
-Contenu_jt : Text
+Envoyer_m()
+Recevoir_m()
+Supprimer_m()
+Consulter()
+Notification()
Publie
Visibilité
-Droit : boolean
Fait
Historique
-date_hi : Date
-heure_hi : int
check-in
-Id_ck : int
-Appréciation : int
+Notif_Ch()
1
*
0..*
*
Publication
-Id_pb : int
-Contenu : Text
+Ajouer_pb()
+Modifier()
+Consulter()
+Supprimer()
+Notif_Pb()
Commentaire
-Id_Cm : int
-Contenu_cm : Texte
-Date_cm : Date
+Ajouter_Cm()
+Modifier_Cm()
+Supprimer_Cm()
*
Critère
-Appréciation : int
possède
1..*
1
Photo
-Id_ph : int
-extension : String
-Contenu : text
+Ajouter()
+modifier()
+Consulter()
+Supprimer()
+Notif_pb()
Video
-Id_vd : int
-extension : String
-Contenu : Text
+Ajouter()
+Modifier()
+Consulter()
+Supprimer()
+Notif_pb()
Statut
-Id_st : int
-Contenu_st : texte
+Ajouter()
+Modifier()
+Consulter()
+Supprimer()
+Notif_pb()

Exercice 2 : Mise en œuvre du système d'information d'une banque

- Nous considérons dans cet exercice le cahier des charges suivant :
- Chaque client peut posséder plusieurs comptes en banque. Un client est défini par un
- numéro, un nom et une adresse.
- Un compte est défini par son numéro et le solde. Deux types de comptes existent, l'un
- autorisant un certain découvert au client, l'autre pas.
- L'application contient aussi une classe *ConvertisseurDollar* qui permet de traduire les
- montants des transactions du Dollar vers l'euro et inversement.
- **Q6:** Un de vos collègues vous fournit le diagramme de classe suivant. Après lui avoir
- notifié les trois erreurs dans ce diagramme, mettre en oeuvre la structure de ce système d'information. Il n'est pas nécessaire de mettre en oeuvre les méthodes de ces classes.


Banque
consultation(in num_compte: String)
conversion_from_euro(in montant: float): float
conversion_to_euro(in montant: float): float
depot(in num_compte: String, in montant: float)
ouverture_compte(in com_client: String)
retrait(in num_compte: String, in montant: float)
- banque
- clients
1
*
Client
adresse: String
nom: String
numeroClient: String
1
- titulaireCompte
*
- comptes
- comptes
1..*
Compte
numeroCompte: String
solde: float
crediter(in montant: float)
debiter(in montant: float)
0..1
- conv
ConvertisseurDollar
fromEuros(in montant: float): float
toEuros(in montant: float): float
CompteSansDecouvert
debiter(in montant: float)
CompteAvecDecouvert
decouvertAutorise: float
debiter(in montant: float)

EXERCICE 3 : TRADUIRE LE CODE JAVA EN DIAGRAMME DE CLASSES

```
public interface Délimitée
{
public Frontière getFrontière ;
}
```

```
abstract public class Forme implements Délimitée
{
}
```

```
public class Cercle extends Forme
{
protected float radius ;
protected Point centre ;
public Point getCenter() ;
public float getRadius() ;
}
```

```
public class Ellipse extends Forme
{
protected float a ;
protected float b ;
protected float angle ;
protected Point centre ;
public Point getCentre() ;
public float getAngle() ;
}
```

```
public class Point extends Forme
{
protected float x ;
protected float y ;
protected float z ;
public float getX() ;
public float getY() ;
public float getZ() ;
}
```

```
abstract public class ListePoints extends Forme
{
protected Arraylist <Point> points = new Arraylist<Point>() ;
public int getComptePoints() ;
public Point getPoint(int i) ;
public ArrayList <Point> getPoints() ;
}
public class Polygone extends ListePoints
{
}
public class Ligne extends ListePoints
{
}
```

```
public class Frontière extends Forme
{
}
public class ListeFormes extends Forme
{
protected Arraylist <Forme> formes = new Arraylist<Forme>() ;
public void ajouterForme(Forme forme) ;
public void supprimerForme(int i) ;
}
```

ANNEXE