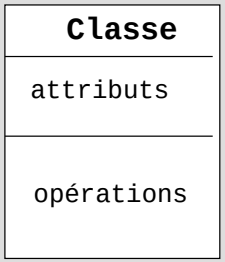
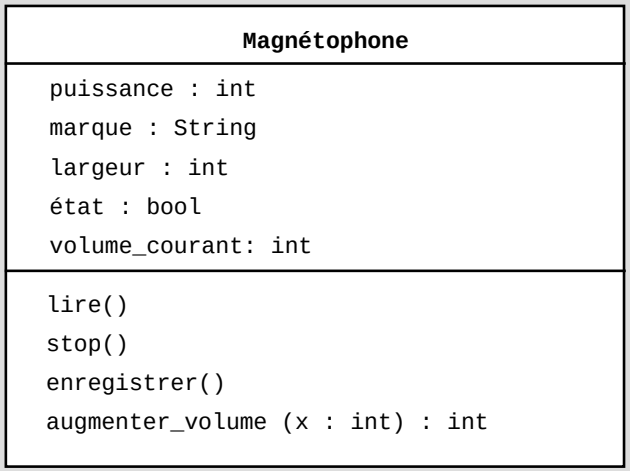
Conception objet

- La conception et la programmation objet consistent à structurer les programmes en objets qui appartiennent à des classes et communiquent entre eux par des envois de messages
  - ➔ Encapsulation
  - ➔ Réutilisation
  - ➔ Tests et maintenance
- Un diagramme de classes (DCA) est une représentation graphique structurée des concepts « métiers » du domaine
- Un objet est caractérisé par :
  - ➔ Ses attributs et son état : ses caractéristiques propres
    `Il possède un OID non visible qui l'identifie`
  - ➔ Ses opérations (ou méthodes) : ses responsabilités

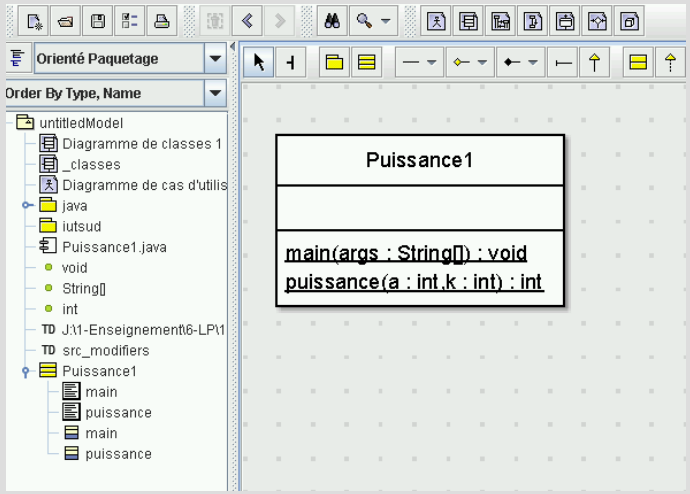
- Classe :
  - ➔ Collection d'objets caractérisés par une sémantique commune : Définition, responsabilités, attributs, opérations, état,...
- Objet
  - ➔ Un élément de la collection (une instance)
- Notation UML



- Exemple UML



- Exemple 2 : (reverse sur le pg « Puissance1 »)



Attributs

- La valeur des attributs est propre à chaque objet
- Peut avoir une forme structurée ou répétitive : prénoms[1.. 3]
  - ➔ *Un objet n'a pas nécessairement d'attribut identifiant, sauf si celui-ci est utilisé dans le monde réel; on peut le mettre en évidence avec l'étiquette {id}*
    Ex : No sécu {id}, N°de commande{id},...
- Syntaxe
  `[visibilite]nom [[multiplicité]][:type][=valeurInit]`
- Les attributs peuvent être typés, avoir une visibilité et/ou une valeur par défaut
  - ➔Ex : # abscisse : int = 0
- Taduction java

  déclaration : `protected int abscisse=0;`
  référence : `objet.attribut`

Opération :

- ➔Service rendu par la classe ; fait partie de ses responsabilités
- ➔Cette opération peut être utilisée par l'objet lui-même ou par d'autres objets (message)
- ➔Elle peut être autonome ou nécessiter la collaboration d'autres classes

- Syntaxe UML :

```
[visibilite]nom [(liste-param)][:type]
        opération()
        opération(x,y)
        + opération(x:Integer, y:String):int
        » Ex : calculerJour(d:Date): String
```

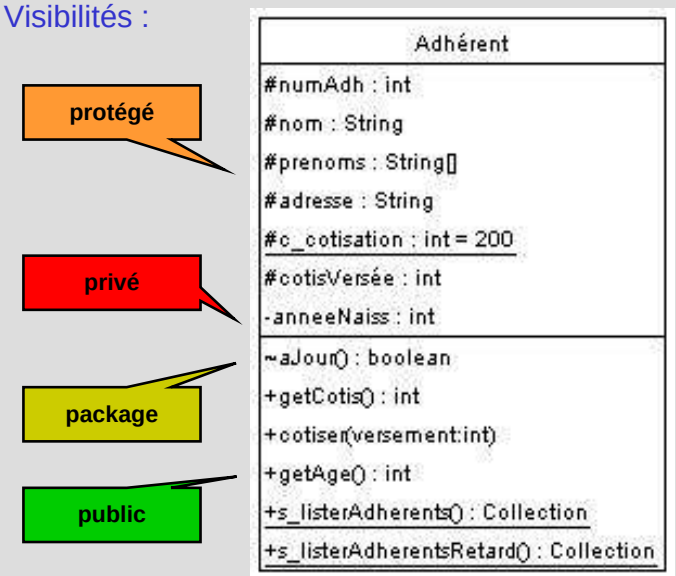
- Traduction Java

```
Déclaration :public int operation(int x, String y)
Appel : objet.methode(parametres)
» Ex : String j=calculerJour(Date today);
```


Visibilités

- ➔ Privé : Accessible seulement depuis la classe elle-même
  (UML : - ; Java : `private`)
- ➔ Package : Accessible seulement depuis les classes du même package
  (UML : ~ ; Java : rien)
- ➔ Protégé : Accessible depuis la classe elle-même, ses dérivées et les classes du même package
  (UML : # ; Java : `protected`)
- ➔ Publique : Accessible depuis toutes les classes.
  (UML : + ; Java : `public`)

Visibilités :



Utilisation des classes en java

- En java, tout est classe (sauf les types de bases)

- Syntaxe Java
  - ➔ Déclaration

```
class Adherent{
  //les attributs de la classe :
  private int numadh;
  private String nom;
  ...etc
  // les operations :
  boolean ajour(){…}
  public int getCotis(){…}
}
```


- Utiliser un objet créé à partir d 'une classe

➔Declaration de la variable

```
» Adherent toto;
```

– *crée une référence nommée toto sur un objet de type Adherent*

– *NB : Pas de création d 'objet à ce niveau*

➔ Création de l 'objet (instance)

```
» toto=new Adherent();
```

– *Crée une instance d 'adherent et lui affecte la référence toto*

- Utiliser un objet créé à partir d 'une classe

➔Constructeur

– *Toute classe possède au moins un constructeur*

– *C 'est l 'opération qui est appelée lors de la création*

– *Un constructeur a le même nom que la classe*

– *Il peut avoir des paramètres*

– *Il initialise généralement les variables*

```
» public Adherent(String n) {...}
» public Adherent(int i,String n) {...}
```

➔ Destructeur

– *Les instances sont détruites (par le garbage collector) dès qu'on ne les utilise plus (= elles ne sont plus référencées)*

➔Déclaration, création, utilisation

```
» Adherent adh1=new Adherent();
» adh1.numadh=123;
» adh1.nom= ''Dumoulin''
» Adherent adh2=new Adherent(x,''Dupont'');
```


- toString()

➔ Chaque classe java possède une méthode « toString() » qui permet d 'afficher ses caractéristiques principales ; c 'est ce qui permet d 'écrire :

```
» JOptionPane jop = new JOptionPane();
» System.out.println(jop);
» ou
» int i;
» System.out.println('' i = ''+i);
```

fait un appel implicite à la méthode « toString » :

```
» System.out.println(jop.toString());
```

➔ Prenez l'habitude de créer une telle méthode pour les classes que vous développez

– *Affichage*

– *débuggage*

- Le concept de CLASSE en modélisation objet revêt trois aspects simultanément en phase d'analyse :
  - ➔La classe est un concept
  - ➔La classe est un modèle
  - ➔La classe est un ensemble

➢Concept

- La classe représente un concept métier, il appartient à l'univers du discours
  - Ex : La classe «Adherent» représente le concept d 'adhérent
- La pertinence de ce concept permettra d'assurer une bonne communication entre l'utilisateur et le développeur
  - ***«Choisissez les noms avec soin. Utilisez des noms de classes significatifs, choisis dans le domaine et connus par les utilisateurs et les experts du domaine» J. Rumbaugh.***

➢Modèle (type abstrait)

- C 'est l 'aspect programmation objet
- La CLASSE est un modèle, un moule servant à la création d'un nouvel OBJET : chaque objet est une instance de la classe
- Chaque objet créé possèdera les mêmes méthodes (celles définies dans la classe) et les mêmes attributs mais pas les mêmes valeurs d 'attributs !
  - ➔Ex : Une classe « Adherent » créera des instances d 'adhérents ayant tous la même structure mais des noms, adresses différents

➢Ensemble

- Une classe peut aussi être considérée comme l'ensemble des objets qui :
  - ➔répondent au concept
  - ➔ont été créés sur son modèle
- cf concept d 'entité dans le modèle entité / association
- Ex : La classe « Adherent » représente en analyse l 'ensemble des adhérents qui ont été créés
- En tant qu 'ensemble, elle possède des propriétés de classe (attributs ou opérations ensemblistes)

Propriétés de classe

- Il est utile de distinguer, lors de l 'analyse, 3 sortes de propriétés :

Individuelles, collectives, communes

➔Les propriétés collectives et communes sont appelées «de classe»

Propriétés individuelles

➔ Attribut ou méthode propre à (et pertinente pour) une instance de la classe
- *ex : le nom, l 'âge de l 'adhérent*

➔ On accède à ces propriétés par la référence de leur instance

```
» System.out.println(x.nom);
```

➔C 'est le cas le plus courant

Propriétés collectives

➔ Attribut propre à la classe en tant qu'ensemble ou opération pertinente seulement sur l'ensemble des instances

➔ En UML, ces propriétés sont soulignées et préfixées par «s»

```
Ex : s-nombreAdherents
s-calculerNbAdherentsEnRetard()
```

➔ En java, ces propriétés sont déclarées « static », l 'accès à ces propriétés n 'est pertinent que par la classe et non par ses instances

Déclaration : `static type s-attribut;`
Reference : `Classe.s-attribut`
Déclaration : `static type s-methode();`
Appel : `Classe.s-methode();`

```
System.out.println(Adherent.s-nombreAdherents);
System.out.println(Adherent. s_calculerNbAdherentsEnRetard();
```


Propriétés communes

➔C'est un attribut commun à toutes les instances de la classe ou une opération pertinente à la fois pour la classe comme ensemble et pour chaque instance
- *Le nombre de pattes pour la classe « Insecte »*
- *Le taux de TVA pour la classe « PiecesDetachees »*

➔En UML, ces propriétés sont soulignées et préfixées par «c»

➔Declaration : `c-nbDePattes=6` *dans la classe* `«Insecte»`

➔ En java, ces propriétés sont aussi déclarées « static », l 'accès à ces propriétés est pertinent par la classe et par ses instances

Déclaration : `static int c-nbDePattes;`

➔ On peut y accéder par chaque instance

```
Insecte fourmi= new Insecte();
System.out.println (fourmi.c-nbDePattes) ;
```

➔ Ou par la classe :

```
System.out.println (Insecte.c-nbDePattes) ;
```

Propriétés de classes (exercice)

Les classes UML et Java

Dans l'objet «client» quel est le type (I,C,S) de chaque propriété?

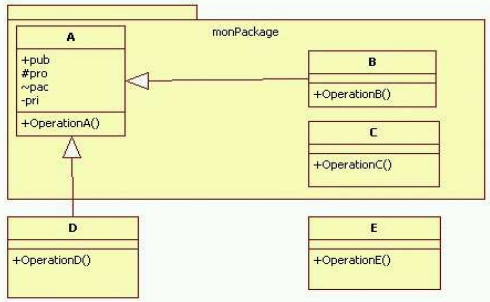
année_naissance
calculer_age()
code_client
prochain_code_client
afficher_caracteristiques (code_cli)
afficher_caracteristiques ()
nb_total_de_clients
taux_de_remise
taux_de_remise_max
afficher_moyenne_d_age( )
quel_code_client ('Dupont')
affiche_nom_du_client( )
ajouter_nouveau_client(1234,'Durand',1958)

CLIENT

exercice

Les classes UML et Java

1.8 Soit le diagramme suivant :



Précisez sur le tableau suivant quels sont, parmi les 4 attributs de la classe A, ceux qui sont accessibles depuis les opérations de chaque classe. (mettez une croix si l'attribut est accessible)

	pub	pro	pac	pri
operationA()				
operationB()				
operationC()				
operationD()				
operationE()				