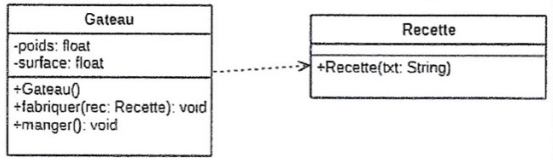
1) Les dépendances

- Exemple complet en Java :




```
public class Appli {

      public static void main(String[] args) {
            String txt = "100g chocolat; 3 oeufs";
            Recette maRecette;
            Gateau unBeauGateau;

            maRecette = new Recette(txt);
            unBeauGateau = new Gateau();
            unBeauGateau.fabriquer(maRecette);

            unBeauGateau.manger()

      }
}
public class Recette {

      public Recette(String txt) {
            System.out.println("Ceci est la recette " + txt);
      }

}
```

points d'arrêts

```
public class Gateau
{
      private float  poids;
      private float surface;
      public Gateau()
      {

      }
      public void fabriquer(Recette rec )
      {
            System.out.println("J'utilise la recette");
      }
      public void manger()
      {
            System.out.println("C'est plutôt bon");
      }
}
```

Arrêt programme

```
Ceci est la recette100g chocolat; 3 oeufs
J'utilise la recette
```

no method return value

this		Gateau (id=21)
poids		0.0
surface		0.0

Rec n'existe plus

2) Les associations simples

- Exemple 1 (Java) : association bidirectionnelle

employe est une propriété
employeur est une propriété

```
class Personne
{
    private string nom;
    private string prenom;
    private Entreprise employeur;
}
```

```
class Entreprise
{
    private string nom;
    private string adresse;
    private Personne[] employe;
}
```


2) Les associations simples

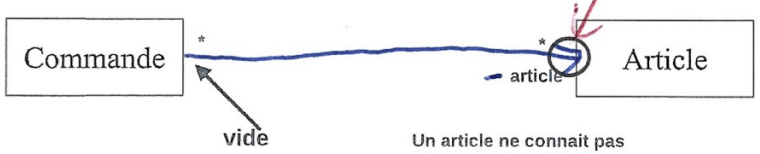
- Exemple 2 (Java) :


```
class Commande
{
    private Date date;
    Private Article[] article;

}
```

```
class Article
{
    private String nom;
    private String adresse;
    Private Commande[] commande;

}
```


2) Les associations simples

- Exemple 3 (Java) :



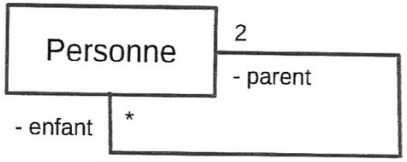

```
class Commande
{
    private Date date;
    Private Article[] article;

}
```

```
class Article
{
    private String nom;
    private String adresse;
    Private Commande[] commande;
}
```

car Commande n'existe plus

3) Les associations réflexives

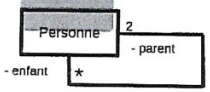
- Exemple 1 (Java) :




```
class Personne
{
    private String nom;
    private String prenom;
    private Date dNaissance;
    private List<Personne> enfant = new ArrayList<Personne>();
    private Personne[] parent = new Personne[2];
}
```


3) Les associations réflexives

- Exemple 1 (Java) :




```
public class Application {

public static void main(String[] args) {
      Personne p1,p2,p3,p4;

  p1 = new Personne("Supormoi","Steven","13/04/1999");
  p2 = new Personne ("Passepartout", "Chris","10/12/2002");
  p3 = new Personne("Cetrouilllard","Stéphane", "15/06/1960");
  p4 = new Personne("Jort","Danny","01/01/1958");

  p3.avoirEnfant(p1);
  p3.avoirEnfant(p2);
  p1.definirParents(p3, p4);
  p1.sePresenter();
      }
}
```

```
public class Personne {
      private String nom;
      private String prenom;
      private String dNaissance;

      private  List<Personne> enfant;
      private Personne[] parent = new Personne[2];

      Personne(String nom, String prenom, String dNaissance)
      {
            this.nom = nom;
            this.prenom = prenom;
            this.dNaissance = dNaissance;
            enfant = new  ArrayList<Personne>();
      }
      public void definirParents(Personne p1, Personne p2)
      {
            parent[Genre.PERE] = p1;
            parent[Genre.MERE] = p2;
      }
      public void avoirEnfant(Personne p1)
      {
            enfant.add(p1);
      }
      public void sePresenter()
      {
            System.out.println("Je m'appelle "+prenom + " "+nom);
            System.out.println("mon père c'est "+parent[Genre.PERE].prenom
            + " "+parent[Genre.PERE].nom);
            System.out.println("ma mère c'est "+parent[Genre.MERE].prenom
            + " "+parent[Genre.MERE].nom);
      }
}
```

this	Personne (id=21)
dNaissance	"15/06/1960" (id=23)
enfant	ArrayList<E> (id=30)
[0]	Personne (id=22)
[1]	Personne (id=45)
nom	"Cetrouilllard" (id=39)
parent	Personne[2] (id=40)
prenom	"Stéphane" (id=41)

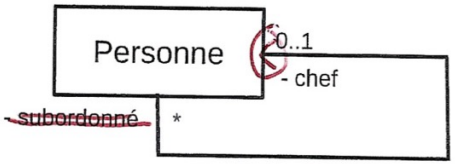
this	Personne (id=22)
dNaissance	"13/04/1999" (id=23)
enfant	ArrayList<E> (id=30)
nom	"Supormoi" (id=39)
parent	Personne[2] (id=40)
[0]	Personne (id=44)
[1]	Personne (id=21)
prenom	"Steven" (id=41)

```
Je m'appelle Steven Supormoi
mon père c'est Stéphane Cetrouilllard
ma mère c'est Danny Jort
```

Genre
+MERE: int = 0
+PERE: int = 1

3) Les associations réflexives

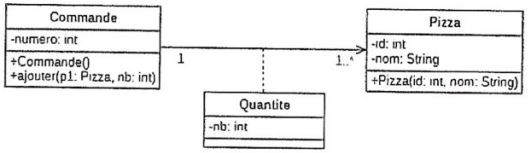
- Exemple 2 (Java) :




```
class Personne
{
    private String nom;
    private String prenom;
    private Date dNaissance;
    private List<Personne> subordonne = new ArrayList<Personne>();
    private Personne chef = new Personne();
}
```


4) Les classes d'association

- Exemple 1 (Java) :




```
public class Commande {
      private int numero;
      private Map<Pizza, Integer> pizzas;

      Commande()
      {
            pizzas = new HashMap();
      }
      public void ajouter(Pizza p1, int nb)
      {
            pizzas.put(p1, nb);
      }
}
```

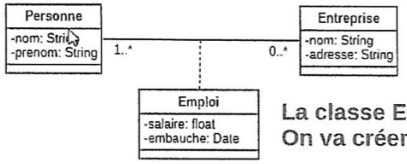
```
public class Pizza {
      private int id;
      private String nom;

      public Pizza(int id, String nom)
      {
            this.id = id;
            this.nom = nom;
      }
}
```

La classe Quantité est assez basique. On ne va pas créer de classe en Java

La classe HashMap contient les paires <clé, valeur>.
Cela permet de gérer de façon simple une quantité de pizzas

4) Les classes d'association

- Exemple 2 (Java) :



La classe Emploi est complexe.
On va créer une classe en Java

```
class Personne
{
    private String nom;
    private String prenom;
    private list<Emploi> emploi;
}
```

```
class Entreprise
{
    private String nom;
    private String adresse;
    private list<Emploi> emploi;
}
```

```
class Emploi
{
   private Personne employe;
   private Entreprise employeur;
   private Date embauche;
   private float salaire;
}
```

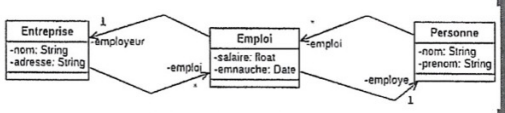
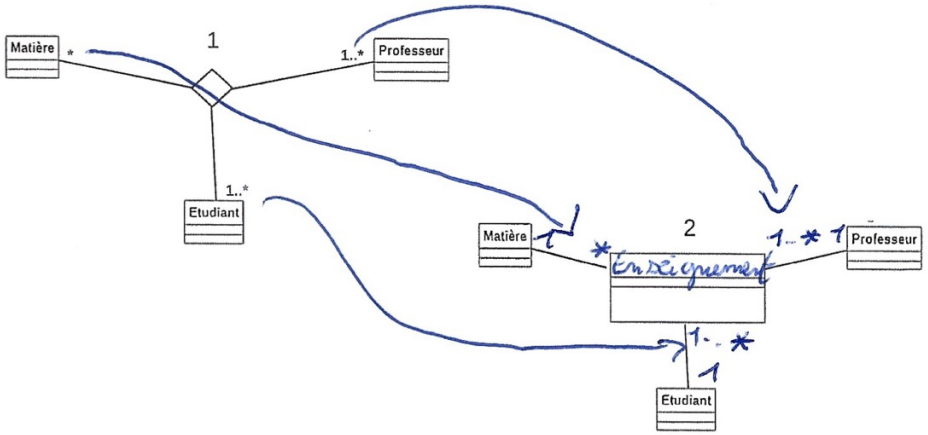


Schéma équivalent

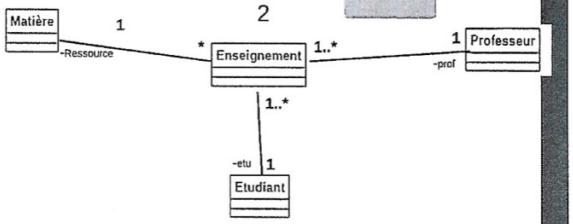
5) Les associations n-aire

- Exemple (Java) :



5) Les associations n-aire

- Exemple 2 (Java) :




```
class Etudiant
{
   private list<Enseignement> ens;
   ...
}
```

```
class Enseignement
{
    private Etudiant etu;
    private Professeur prof;
    private Matiere ressource;
    ...
}
```

```
class Professeur
{
   private list<Enseignement> ens;
}
```

```
class Matiere
{
   private list<Enseignement> ens;
}
```


6) Les agrégations

- Exemple 1 (Java) :

Les personnes appartiennent à une équipe mais existent même sans l'équipe

```
public class Personne {

    private String nom;
    private String prenom;
    public Personne(String nom,String prenom) {
      nom = this.nom;
      prenom = this.prenom;
    }
}
```

```
public class Application {
  public static void main(String[] args) {
    Personne p1,p2,p3,p4;
    Equipe teamRequete;

    p1 = new Personne("Supormoi","Steven");
    p2 = new Personne("Super","Didier");
    p3 = new Personne("Pasamsung","Christelle");
    teamRequete = new Equipe();
    teamRequete.ajouterMembre(p1);
    teamRequete.ajouterMembre(p2);
    teamRequete.ajouterMembre(p3);
    teamRequete.retirerMembre(p1);
    System.out.println("Il y a "+teamRequete.lireEffectif()+"
personnes dans l'équipe" );
  }
}
```

```
public class Equipe {

    private String nom;
    private int effectif;
    private List<Personne> membre
     = new ArrayList<Personne>();

    public Equipe() {
      effectif = 0;
    }

    public int lireEffectif() {
        return effectif;
    }

    public void ajouterMembre(Personne m) {
      membre.add(m);
      effectif++;
    }

    public void retirerMembre(Personne m) {
      membre.remove(m);
      effectif--;
    }
```

```
Problems  Javadoc  Declaration
<terminated> Application (3) [Java Applicatio
Il y a 2 personnes dans l'équipe
```

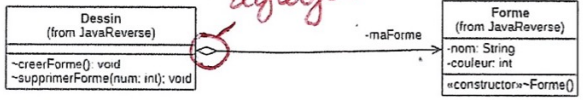
R2.01 Développement orienté objets

Cours N°4 : UML : Les diagrammes de classes : les associations

Les agrégations

6) Les agrégations

- Exemple 2 (Java) :



agrégat°

```
public class Dessin
{
      private List<Forme> maForme = new ArrayList<Forme>();
      void creerForme()
      {
            Forme f = new Forme();
            maForme. add(f);
      }
      void supprimerForme(int num)
      {
            maForme. remove(num);
      }
}
```

Les formes appartiennent au dessin

```
public class Forme {
      private String nom;
      private int couleur;

      Forme()
      {
            System.out.println("Nouvelle forme");
      }
}
```

Le dessin peut créer ou supprimer Des formes

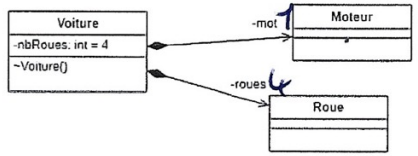
R2.01 Développement orienté objets

Cours N°4 : UML : Les diagrammes de classes : les associations

Les composition

7) Les compositions

- Exemple 1 (Java) :




```
public class Voiture {

      private Moteur mot;
      private Roue[] roues;
      private int nbRoues = 4;

      Voiture()
      {
            mot = new Moteur();
            roues = new Roue[nbRoues];
            for (int i=0; i<nbRoues; i++)
                  roues[i] = new Roue();
      }
}
```

La voiture n'existe qu'avec son moteur et ses roues.

On pourra quand même en changer